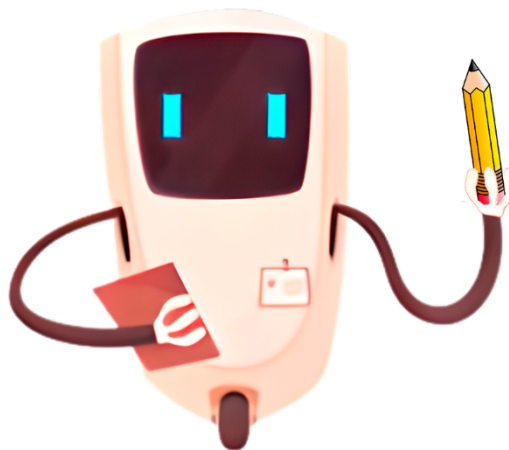


Mee, gestor personal

Grado Superior Desarrollo de Aplicaciones Web

Alumno: Alejandro Ruiz Alfás
Tutor: Juan Segura Vasco



Curso 2022-2023

14 junio 2023

Agradecimientos

A mis padres y mi hermana, por todo el apoyo que me han otorgado, no solo monetario sino también afectivo. Gracias por confiar siempre en mí y gracias por ayudarme a financiar mi futuro, tanto académico como profesional.

A mis profesores, tanto del grado medio como del grado superior, por realizar su trabajo con vocación y enseñarme todo aquello que hoy en día puedo aportar al mundo laboral, por tener la paciencia que han tenido con nosotros, y por encima de todo, por enseñarnos a ser mejores personas.

A mis amigos por forjar una familia en tan solo 4 años desde que nos conocimos en el grado medio. Por todas las risas y discusiones que hemos tenido y por todos los buenos y malos momentos que día a día hemos vivido hasta hoy y por ayudarnos entre todos con cualquier cosa. Gracias por todo, “soldados caídos”.

A mis compañeros de clase del grado superior, por compartir este año juntos, con las risas y bromas en clase, y por hacer que estudiar todos los días pueda ser no solo costoso, sino también divertido.

A mi instructor y mis compañeros de trabajo por hacer que el periodo de prácticas en centros de trabajo no sea tan solo trabajar, sino que sea mucho más ameno y divertido, además de brindarme con un día semanal para avanzar en mi proyecto y memoria.

Contenido

Agradecimientos	2
1.Introducción	5
1.1. Origen y etimología	5
1.2 ¿Qué o quién es Mee?	5
1.3 Usos de la aplicación	6
2. Antecedentes	6
2.1 Origen de las aplicaciones web	6
2.2 JavaScript. ¿Qué es y cómo nace?.....	7
Orígenes y Evolución:	7
Frameworks y Bibliotecas:.....	8
Actualidad:	8
Conclusión:	8
3. Marco teórico	9
3.1 Vue vs React vs Angular vs Svelte.....	9
4. Funcionamiento.....	10
4.1 Home	10
Explicación general.....	11
Explicación técnica	12
4.2 Todo-List (Lista de cosas para hacer)	13
Explicación general.....	13
Explicación técnica	15
4.3 Notas y recordatorios	16
Explicación general.....	16
Explicación técnica	17
4.4 Buylist (Listas de la compra)	19
Explicación general.....	19
Explicación técnica	21
4.5 Map Gallery (Galería de Mapas)	22
Explicación general.....	22
Explicación técnica	23
4.6 Weather (El Tiempo)	25
Explicación general.....	25
Explicación técnica	26
4.7 News (Noticias).....	28
Explicación general.....	28

Explicación técnica	29
4.8 Words (Palabras)	31
4.8.1 Word of The Day (Palabra del día).....	31
Explicación general.....	31
Explicación técnica	31
4.8.2 Translator (Traductor).....	32
Explicación general.....	32
Explicación técnica	33
4.8.3 Meectionary (Diccionario).....	34
Explicación general.....	34
Explicación técnica	34
5. Base de Datos	36
5.1 Esquema ER (Entidad-Relación)	37
5.2 Estructuración de tablas.....	38
Tabla “user”	38
Tabla “notes”	38
Tabla “todos”	39
Tabla “maps”	39
Tabla “buylist”	39
Tabla “categories”.....	40
5. Application Programming Interface (API).....	40
5.1 Plan inicial.....	40
5.2 Hosting	40
5.3 Resolución	41
Get.php.....	42
Add.php.....	43
Edit.php	44
Delete.php.....	45
7. Conclusión	46
7.1 Resultados obtenidos	46
7.2 Puntos pendientes.....	46
Compatibilidad web-móvil	47
7.3. Tiempo dedicado y dificultad	48
7.4. Valoración personal y autoevaluación.....	49
8. Bibliografía y Webgrafía	49
9. Recursos utilizados	50

10. Glosario	51
10.1. Siglas.....	51
10.2. Terminología.....	51
11. Resumen del contenido.....	52

1.Introducción

Mee es una aplicación web y móvil de gestión personal que permite al usuario acceder a distintas funcionalidades útiles para del día a día desde un único lugar centralizado. Dentro de la aplicación web, un pequeño robot llamado Mee servirá de asistente y compañero para el usuario.

1.1. Origen y etimología

La idea de crear a Mee surge en mi cabeza a través de la necesidad de gestionar de forma más sencilla y centralizada todas las tareas y conocimientos de cada día. Desde siempre he sido una persona con buena memoria y planificación, pero incluso con esas cualidades, tenemos demasiadas cosas en la cabeza como para acordarnos de todo en cualquier momento.

Siempre he recurrido a apuntar las cosas sobre papel y bolígrafo, notas del móvil, alarmas y recordatorios... Son métodos viables, pero tener tantos lugares que consultar puede dar lugar a que alguno de ellos quede sin revisar. Es por ello por lo que siempre había soñado con tener algo o a alguien que se encarara de gestionar todas mis recordatorios y notas y me ayudara en mi día a día, pero no parecía tarea fácil...

Así es como nace Mee, mi propio asistente y gestor personal.

La etimología de <<Mee>> es muy sencilla, pues el nombre deriva del pronombre inglés “Me”, cuyo significado y traducción sería “Yo”, pero agregando la segunda letra “e” para darle un toque más simpático y agradable.

1.2 ¿Qué o quién es Mee?

Mee es un pequeño y adorable robot creado con la finalidad de ayudar a las personas. Mee siempre ha sido un robot muy bueno y generoso con la gente, y le encanta ayudar a todo el mundo con sus cosas. Le hace sentir mejor y piensa que ayudar a la gente es una de las principales causas por las que fue creado.

Así pues, ahora Mee te agradece que hayas decidido usar la aplicación web o móvil y como forma de agradecimiento, se ofrece a ayudarte con la gestión de tu tiempo y tus tareas haciendo uso de sus programas internos para conseguirlo. No pide nada a cambio, solo que cada día pases a saludar y juegues con él mientras él se encarga de que no te olvides de nada.

Su software creado especialmente para ayudarte cuenta con diversas funcionalidades como notas, gestión de tareas a realizar, diccionario, listas de la compra, noticias actuales... ¡Seguro que no te faltará de nada!

1.3 Usos de la aplicación

La aplicación está pensada para que cualquier persona pueda usarla y sacar provecho de ella. No está enfocada a un nicho o grupo de personas en concreto, pues la aplicación es de carácter general y cualquier persona del mundo tiene tareas y cosas que hacer, y necesita gestionarlas de alguna forma sencilla.

Debido a la forma en la que está planteada, no solo permite a la persona tener un robot gestor personal, sino que también actúa como una herramienta multiusos, ya que cuenta con diferentes funciones útiles y portables como diccionario, traductor, etc.

Está enfocada a un entorno individual y personal, y no directamente a un uso empresarial o masivo, aunque en este último caso quizá se podría adaptar para que también funcionara en este tipo de entornos.

2. Antecedentes

Para entender mejor el proceso de planteamiento y desarrollo de Mee-Web, es interesante conocer los orígenes de las primeras aplicaciones web, así como la historia de JavaScript y Vue, el lenguaje de programación y framework respectivamente, usados para el desarrollo.

2.1 Origen de las aplicaciones web

El origen de las aplicaciones web en Internet ha sido un hito clave en el desarrollo de la tecnología y la forma en que interactuamos en la era digital. Estas aplicaciones, también conocidas como aplicaciones basadas en la web o simplemente "web apps", han revolucionado la manera en que utilizamos la información y accedemos a servicios en línea.

El concepto de aplicaciones web se remonta a los primeros días de Internet, cuando la World Wide Web comenzó a tomar forma en la década de 1990. Antes de eso, la mayoría de las interacciones en línea se basaban en el correo electrónico y los protocolos de transferencia de archivos. Sin embargo, con la llegada de la web, surgieron nuevas posibilidades de desarrollo de aplicaciones que aprovechaban la infraestructura global de Internet.

Las aplicaciones web se construyen utilizando lenguajes de programación web como HTML, CSS y JavaScript. Estos lenguajes permiten a los desarrolladores crear interfaces de usuario interactivas y funcionales que se ejecutan en un navegador web. A diferencia de las aplicaciones de escritorio tradicionales, las aplicaciones web no requieren instalación en el dispositivo del usuario y se pueden acceder a través de un navegador web en cualquier sistema operativo.

Una de las primeras aplicaciones web famosas fue el navegador web Mosaic, desarrollado en 1993. Mosaic permitía la visualización de páginas web con texto y gráficos, y fue un gran avance en la forma en que se consumía y se compartía información en línea. Esta innovación fue el motivo principal por el cual se empezaron a desarrollar aplicaciones web más complejas en los años siguientes.

A medida que la tecnología web evolucionaba, también lo hacían las aplicaciones web. Surgieron aplicaciones de correo electrónico basadas en la web, como Hotmail y Yahoo Mail, que permitían a los usuarios acceder a sus bandejas de entrada desde cualquier lugar con una conexión a Internet. Luego vinieron las aplicaciones de búsqueda, como Google, que revolucionaron la forma en que encontramos información en línea.

El desarrollo de aplicaciones web continuó acelerándose con la introducción de tecnologías como AJAX (Asynchronous JavaScript and XML), que permitía a las aplicaciones web actualizar contenido en tiempo real sin necesidad de recargar la página completa. Esto dio lugar a la creación de aplicaciones web más dinámicas y responsivas, como Google Maps y Gmail.

En la actualidad, las aplicaciones web son una parte fundamental de nuestras vidas en línea. Desde redes sociales como Facebook y Twitter, hasta plataformas de streaming de música y video como Spotify y Netflix, estas aplicaciones nos brindan una amplia gama de servicios y experiencias interactiva, sin necesidad de descargar nada para su uso y totalmente en línea.

En resumen, el origen de las aplicaciones web en Internet se remonta a los primeros días de la web en la década de 1990. A través de avances tecnológicos y el desarrollo de lenguajes de programación web, las aplicaciones web han transformado la forma en que accedemos a la información y utilizamos servicios en línea. Continúan evolucionando y desempeñan un papel fundamental en nuestra sociedad digital y cada vez toman más y más peso en nuestra vida.

2.2 JavaScript. ¿Qué es y cómo nace?

El lenguaje JavaScript es un lenguaje de programación ampliamente utilizado en el desarrollo de aplicaciones web. Se utiliza principalmente para agregar interactividad y funcionalidad a las páginas web, permitiendo a los desarrolladores crear experiencias dinámicas para los usuarios.



Orígenes y Evolución:

JavaScript fue creado por Brendan Eich en 1995 mientras trabajaba para Netscape Communications. En ese momento, Netscape estaba desarrollando su navegador web Netscape Navigator y necesitaba un lenguaje de programación que pudiera ser ejecutado directamente en el navegador del usuario. Este lenguaje fue llamado inicialmente "LiveScript", pero más tarde se renombró como "JavaScript" para capitalizar el crecimiento de popularidad de Java en ese momento.

Inicialmente, JavaScript se diseñó como un lenguaje de scripting ligero con capacidades limitadas. Sin embargo, a medida que Internet y la web evolucionaron, JavaScript también se expandió para incluir características más poderosas y se convirtió en un estándar de facto en el desarrollo web.

Frameworks y Bibliotecas:

Con el tiempo, se han desarrollado numerosos frameworks y bibliotecas en JavaScript para facilitar el desarrollo de aplicaciones web. Estas herramientas proporcionan conjuntos de funciones y abstracciones que simplifican tareas comunes y aceleran el proceso de desarrollo.

Uno de los frameworks más populares es Angular, desarrollado por Google. Angular es un framework de desarrollo de aplicaciones web de código abierto que utiliza JavaScript para crear aplicaciones de una sola página (SPA) altamente interactivas. Proporciona una estructura sólida para el desarrollo y ofrece una amplia gama de características y herramientas.

Otro framework destacado es React, desarrollado por Facebook. React se centra en la construcción de interfaces de usuario y se utiliza para crear componentes reutilizables y muy eficientes. React utiliza un enfoque basado en componentes y utiliza JavaScript como su lenguaje principal.

Actualidad:

En la actualidad, JavaScript sigue siendo uno de los lenguajes de programación más utilizados en el desarrollo web. Ha evolucionado enormemente a lo largo de los años y ahora se ejecuta tanto en el lado del cliente (navegador) como en el lado del servidor (Node.js).

Además de los frameworks mencionados anteriormente, han surgido otras bibliotecas populares como Vue.js, Ember.js, React.js, Angular.js, Backbone.js, etc., cada una con su propio enfoque y características únicas. Estas herramientas permiten a los desarrolladores crear aplicaciones web más rápidas, escalables y fáciles de mantener.

Conclusión:

JavaScript ha desempeñado un papel fundamental en el desarrollo de aplicaciones web, desde sus modestos comienzos hasta convertirse en un lenguaje poderoso y versátil. Su evolución ha estado marcada por el surgimiento de frameworks y bibliotecas que han facilitado el desarrollo de aplicaciones web más sofisticadas. En la actualidad, JavaScript sigue siendo un lenguaje imprescindible en el ámbito del desarrollo web y se espera que siga creciendo y evolucionando en el futuro.

3. Marco teórico

Hoy en día, para el desarrollo de una aplicación web se pueden utilizar muchas tecnologías y metodologías diferentes, y cada desarrollador escoge un planteamiento y solución distinta para cada problema. Para un mismo lenguaje de programación como JavaScript, existen muchas formas diferentes de plantear una solución a un problema de desarrollo. Se puede hacer uso de algún framework de los indicados en los antecedentes. Unos de ellos son más sencillos de aprender, como Vue o Svelte, mientras que otros pueden ser más completos, pero también más complejos, como Angular o React, entre otros.

También existe la posibilidad de usar Javascript nativo, sin necesidad de depender de un framework. Esto haría que el desarrollo fuese más fiel al lenguaje de programación que se utiliza, e incluso más fluido que si se utilizase un framework, pero la cantidad de trabajo a realizar también aumentaría, ya que los frameworks vienen preparados con una serie de funciones especialmente diseñadas para facilitar al desarrollador muchas de las tareas que, de otra forma, costarían mucho tiempo, pero que estos frameworks ya tienen implementadas y permiten ahorrar mucho tiempo y líneas de código a cambio de una mínima bajada de rendimiento en algunos casos.

A continuación, se muestra una comparativa entre los diferentes frameworks más populares de Javascript en la actualidad.

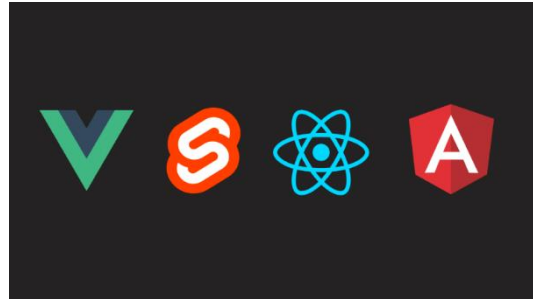
3.1 Vue vs React vs Angular vs Svelte

Vue.js es un framework de JavaScript altamente flexible y fácil de aprender, lo que lo convierte en una excelente opción para desarrolladores principiantes. Su enfoque basado en componentes permite la creación de interfaces de usuario reutilizables, promoviendo así la modularidad del código. Con una documentación completa y una comunidad activa, Vue ofrece un soporte confiable y una curva de aprendizaje suave, lo que facilita el desarrollo de una aplicación web interactiva y eficiente.

React, desarrollado por Facebook, se destaca por su enfoque en la construcción de interfaces de usuario eficientes y dinámicas. A través de su concepto de Virtual DOM, React optimiza el rendimiento de la aplicación al actualizar solo las partes necesarias de la interfaz. Además, React cuenta con un vasto ecosistema de bibliotecas y herramientas, lo que facilita la creación de aplicaciones web complejas y escalables. Su comunidad activa y su adopción extendida en la industria garantizan un amplio respaldo y recursos disponibles.

Angular, desarrollado por Google, es un framework completo y poderoso para el desarrollo de aplicaciones web de mayor envergadura. Utilizando TypeScript, un lenguaje que añade características de tipado estático a JavaScript, Angular mejora la calidad del código y ayuda a prevenir errores. Con una arquitectura basada en componentes y una variedad de características integradas, como enrutamiento y gestión de formularios, Angular se destaca en aplicaciones empresariales. Aunque su curva de aprendizaje puede ser más pronunciada, el conjunto de herramientas y la estructura robusta de Angular hacen de él una elección sólida para proyectos complejos y de gran escala.

Svelte, un framework relativamente nuevo, ha ganado popularidad rápidamente por su enfoque innovador. A diferencia de otros frameworks, Svelte no utiliza una biblioteca en tiempo de ejecución, lo que resulta en un código JavaScript más optimizado y liviano. Esto se traduce en un rendimiento excepcional durante la ejecución de la aplicación.



Con una sintaxis sencilla y una curva de aprendizaje rápida, Svelte es una opción adecuada para proyectos más pequeños y aplicaciones interactivas con menos complejidad.

En conclusión, es fundamental considerar los requisitos específicos y las características del proyecto. Vue.js y React son opciones populares, con comunidades activas y recursos abundantes. Angular destaca en aplicaciones empresariales y de mayor complejidad, mientras que Svelte ofrece un enfoque novedoso y un rendimiento excepcional para proyectos más pequeños. Evaluar cuidadosamente las necesidades del proyecto garantizará una elección acertada para el desarrollo exitoso de la aplicación web.

4. Funcionamiento

A continuación, se van a explicar cada una de los componentes o funcionalidades principales de los que se compone la aplicación “Mee Web”.

De cada una de las funcionalidades, se describirá brevemente con una pequeña descripción general, seguido de una explicación más general, donde se describen los diferentes elementos visuales principales de cada funcionalidad, y la función de cada uno de esos elementos, o como hacer funcionar ciertos elementos para obtener el resultado deseado.

Por último, se realizará una explicación más técnica de cada funcionalidad, donde se explique de forma más concreta cómo funciona realmente cada funcionalidad, con que tecnologías se ha desarrollado, se explicarán las líneas de código más importantes de cada funcionalidad, y se explicarán tecnologías externas en caso de usarlas, como API's externas, entre otras.

4.1 Home

Lo primero que ve un usuario al acceder a una página web o incluso una aplicación móvil, suele ser una pantalla de home o pantalla principal. En esta pantalla suelen haber accesos directos, mensajes de bienvenida, información de uso de la página, etc.

En el caso de Mee, la pantalla Home sirve como menú principal para acceder al resto de funcionalidades que ofrece la aplicación web.

Explicación general

Inicialmente, la pantalla Home iba a ser un montón de accesos directos, tal y como funciona en la versión móvil, pero esto fue descartado, ya que los accesos directos de la aplicación móvil no funcionan en la aplicación web, pues las aplicaciones del teléfono no están instaladas en el ordenador.

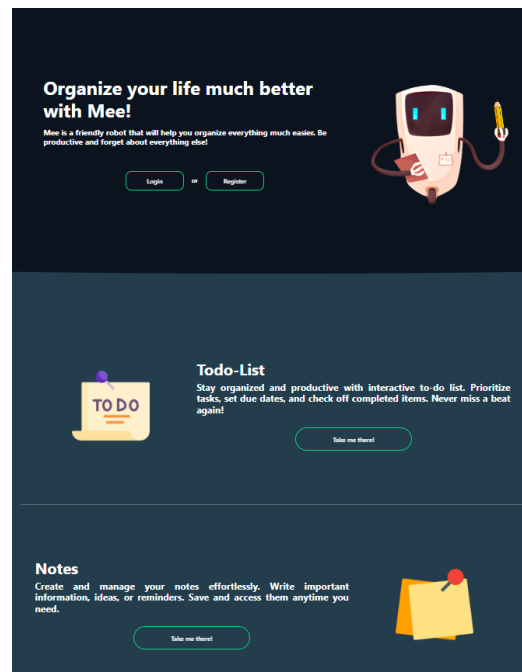
Es por ello que decidí cambiar el funcionamiento de la pantalla Home. Durante el desarrollo pasó por varias fases, pues no estaba muy seguro sobre qué poner en la pantalla Home para que fuese útil.

Al principio, elegí crear accesos directos, pero en vez de a aplicaciones de móvil, esta vez los accesos directos guardarían información sobre las páginas que el usuario quería guardar. Esta no era una mala idea del todo, pero al final es una funcionalidad que existe en todos los navegadores como “Bookmarks” y además obtener la información de la página web que introducía el usuario era más complicado de lo que esperaba, pues no siempre se puede extraer la imagen y título de algunas páginas web por políticas de seguridad.

La idea final fue crear una “landing page” o página de aterrizaje, que como su nombre indica, da la bienvenida al usuario cuando este accede a la web a través del navegador.

La página es muy sencilla y está formada por lo siguientes elementos:

- **Bienvenida y autenticación:** En esta parte hay un mensaje de bienvenida y una imagen de nuestro robot Mee, dándole la bienvenida al usuario y ofreciéndole la posibilidad de iniciar sesión o registrarse en la página.
- **Menú y accesos directos:** En la parte inferior y resto de la Home, se encuentran diferentes apartados similares, cada uno referenciando a una de las funcionalidades disponibles. Ofrece una breve explicación sobre la funcionalidad y un botón para ir directamente a esa funcionalidad, como si de un menú se tratase, pero ofreciendo más información al usuario, para que sepa cómo funciona la aplicación sin ni siquiera haberla usado anteriormente.



Explicación técnica

La pantalla Home en sí no tiene ninguna parte técnica, pues realmente esta solo muestra una serie de diferentes <div> con información sobre todas las funcionalidades. Todos los diferentes elementos del menú utilizan la misma estructura de Bootstrap tanto en HTML como el CSS.

```
<div class="row feature-row">
  <div class="col-md-8 col-12">
    <h1>News</h1>
    <h4>Stay up to date with the latest headlines and breaking news. Search for news articles,
    <div class="text-center">
      <button class="product-row-button mt-4" @click="goTo('news')">Take me there!</button>
    </div>
  </div>
  <div class="col-md-4 col-12 pr-4 feature-icon-wrapper">
    
  </div>
</div>
<hr class="hr" />
```

Cada una de las tarjetas está formada por un título, siendo este el identificador de la funcionalidad, bajo el título una breve descripción de lo que permite realizar esa funcionalidad en concreto, y bajo del todo un botón, que permite moverse directamente a la funcionalidad que se ha elegido.

Utilizando Bootstrap, la tarjeta se divide en 2 partes, una ocupando 2/3 y la otra el 1/3 restante, de forma que los textos se quedan a la izquierda, mientras que la imagen ocupa el espacio restante a la derecha.

4.2 Todo-List (Lista de cosas para hacer)

Las personas siempre estamos haciendo cosas, y como personas responsables tenemos no una si no varias tareas diferentes para realizar a lo largo del día. Es normal no tener tiempo a realizar todas, o incluso olvidarse de realizar alguna tarea, sin importar la dificultad de esta.

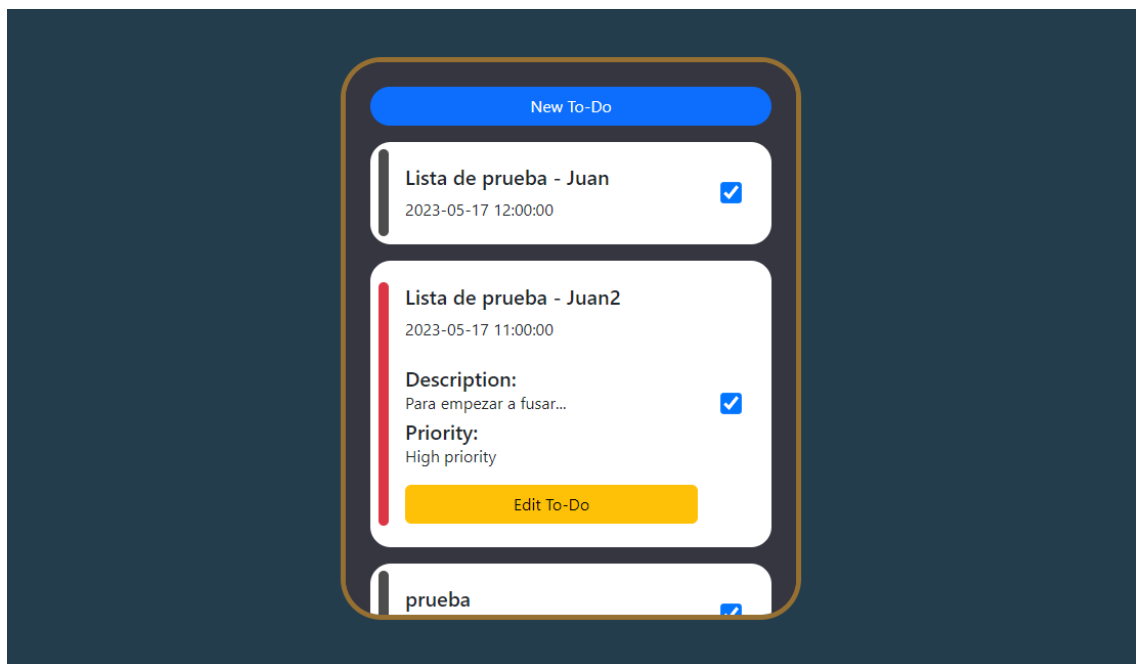
Es por eso por lo que Mee ha planteado la siguiente solución. Una funcionalidad que permita al usuario organizar todas sus tareas en un tablero, donde el usuario puede crear y modificar sus tareas, e incluso establecer prioridades y alarmas a cada una.

Explicación general

La función de lista de cosas para hacer permite al usuario organizar mejor sus tareas, permitiendo crear varias tarjetas con un nombre, una descripción y una fecha de vencimiento. El usuario podrá consultar las tareas que tiene rápidamente e incluso establecer alarmas para la aplicación móvil.

Para ello la interfaz es muy sencilla, pues es tan simple como un tablero formado por dos partes diferentes:

- **Botón de crear tarea:** El botón permite al usuario crear una nueva tarea. Esto abrirá una nueva ventana de creación de tarea explicada más abajo.
- **Lista de tareas:** Debajo del botón se encuentra la lista de las tareas, donde el usuario puede marcar o desmarcar la tarea como completada, permitiendo una organización más cómoda, rápida y efectiva.

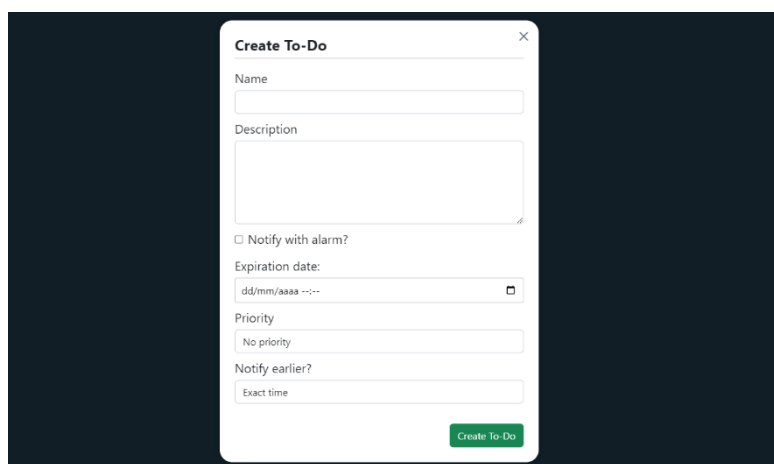


En cuanto a la ventana de agregar tareas, consiste en un formulario básico donde el usuario puede introducir los datos de la nueva tarea, entre ellos el nombre de la tarea, una breve descripción adicional por si se quieren incluir pequeños detalles, la posibilidad de ser notificado en la aplicación móvil, la fecha de vencimiento teórica de la tarea, prioridad y notificación con antelación.

Para editar una tarea, basta con hacer click a la tarea y se mostrará el mismo formulario, pero esta vez con los datos cargados de la tarea que se ha elegido. Los campos del formulario son los mismos, la única diferencia es la posibilidad de pulsar el botón de eliminar para borrar la tarea en caso de haberla completado o haberse equivocado al crear la tarea.

Los campos del formulario, tanto para el caso de añadir una nueva tarea, como el de modificar una tarea existente son los siguientes:

- **Nombre:** El usuario puede indicar el nombre identificativo de la tarea. Este debe de ser corto pero conciso, que permita identificar el objetivo de la tarea a simple vista y sin leer demasiado.
- **Descripción:** El nombre es un texto muy corto, mientras que la descripción permite realizar una breve explicación de la tarea si fuese necesario, o bien incluir pequeños detalles o matices importantes de la tarea.
- **Notificar con alarma:** Esta casilla permite al usuario indicar si quiere que se le notifique del vencimiento de la tarea en la aplicación móvil de Mee.
- **Fecha de vencimiento:** Permite establece al usuario cuál será la fecha máxima para poder completar su tarea.
- **Prioridad:** Establece el nivel de prioridad de la tarea que se va a crear. Este es un concepto teórico, y será el propio usuario quien establezca los límites de la prioridad, según crea conveniente. Existe cuatro opciones de prioridad:
 - **Sin prioridad (color gris):** Indica al usuario que realmente no hay prisa en completar esta tarea, no es necesario hacerla pronto, ni tampoco es muy importante.
 - **Baja prioridad (color verde):** La prioridad de completar esta tarea es baja, por lo que, en caso de existir otras tareas con mayor prioridad, se debería de priorizar completar esas primero.
 - **Media prioridad (color naranja):** La tarea tiene una prioridad media, por lo que hay que tener en cuenta esta tarea e intentar que no se olvide realizarla.
 - **Alta prioridad (color rojo):** La prioridad de la tarea es máxima. El usuario debe completar esta tarea cuanto antes y además dentro del tiempo de vencimiento. ¡No podemos olvidar esta tarea!
- **Notificar con antelación:** Si el usuario activa la notificación por alarma, este campo permite establecer si quiere recibir la alarma justo en el momento de vencimiento de la fecha, 10 minutos antes, 30 minutos antes o 1 hora antes.



Explicación técnica

La funcionalidad de lista de cosas para hacer no es demasiado compleja en cuanto a código, sino más bien en cuanto a maquetación y diseño. No hace llamadas a API externas, sino que sólo utiliza la API propia de Mee para almacenar, consultar y modificar los datos de todas las tareas de los usuarios.

Una vez la aplicación lee todos los todo que tiene el usuario específico almacenados en la base de datos a través de la API, esto se guardan en un array y se muestran en el tablero de la siguiente forma:

- **Datos:** La parte superior del código muestra una barra de color con la prioridad de la tarea (explicada anteriormente en la explicación general) según la prioridad de la tarea, además de mostrar el nombre o título de la tarea, y la fecha de vencimiento.
- **Información adicional oculta:** En la imagen, corresponde a la parte del código marcada en rojo. Esta parte muestra información adicional como la descripción, la prioridad, esta vez en texto y un botón para poder editar la tarea.

```
<div class="todo-item m-3" v-for="item in todos" :key="item.todo_id" @click="openMoreDetails">
  <div v-if="item.priority == 3" class="col-1 todo-prior-bar m-1 mx-2 todo-prior-bar-high"></div>
  <div v-else-if="item.priority == 2" class="col-1 todo-prior-bar m-1 mx-2 todo-prior-bar-medium"></div>
  <div v-else-if="item.priority == 1" class="col-1 todo-prior-bar m-1 mx-2 todo-prior-bar-low"></div>
  <div v-else class="col-1 todo-prior-bar m-1 mx-2 todo-prior-bar-nothing"></div>
  <div class="col-9 todo-content">
    <div class="row">
      <h5 class="my-0 mb-1">{{ item.name }}</h5>
    </div>
    <div class="row">
      <p class="my-0 mt-1">{{ item.limitDate }}</p>
    </div>
    <div class="row more-info-todo mt-4">
      <div class="col-12">
        <h5 class="m-0">Description:</h5>
        <p class="m-0 mb-1">{{ item.desc }}</p>
        <h5 class="m-0">Priority:</h5>
        <p>{{ getPriorityString(item.priority) }}</p>
        <button class="btn btn-warning w-100" @click="handleEditTodo(item)">Edit To-Do</button>
      </div>
    </div>
  </div>
  <div class="col-2 todo-controls">
    <input type="checkbox" @click.stop :checked="item.completed == 1" @click="checkOrUncheck(item)" style="transform: scale(1.6)" />
  </div>
</div>
```

Para mostrar esta información oculta el usuario debe hacer click en cualquiera de las tareas. Esto llamará a un método, el cual se encarga de poner visible todo el <div> usando la clase que tiene asignada, además de crear una pequeña animación para dar la sensación al usuario que se ha desplegado más información de la que antes había.

```
openMoreDetails() {
  if (event.currentTarget.children[1].children[2].style.display === 'block') {
    event.currentTarget.style.transition = 'height 0.5s';
    event.currentTarget.style.height = '100px';
    event.currentTarget.children[1].children[2].style.display = 'none';
  } else {
    event.currentTarget.style.transition = 'height 0.5s';
    let curTarg = event.currentTarget;
    curTarg.style.height = '280px';
    setTimeout(function () {
      curTarg.children[1].children[2].style.display = 'block';
    }, 500);
  }
},
```

Además, desde el botón de editar el usuario puede modificar los datos de cada tarea, así como borrar dicha tarea.

4.3 Notas y recordatorios

Las personas tenemos muchas responsabilidades y cosas que hacer. Muchas de esas cosas son fáciles de recordar, pero muchas otras son pequeños recordatorios que no solemos recordar siempre. Es por eso por lo que Mee ha decidido crear una funcionalidad para que el usuario pueda apuntar todas las cosas que necesite en pequeñas notas de colores.

Esta simple pero amigable funcionalidad permite al usuario escribir pequeñas notas con título y descripción, de forma que estas permanecen almacenadas y permiten al usuario consultar rápidamente. Esta funcionalidad es similar a los To-do, pero sin alertas y mucho más sencilla.

Explicación general

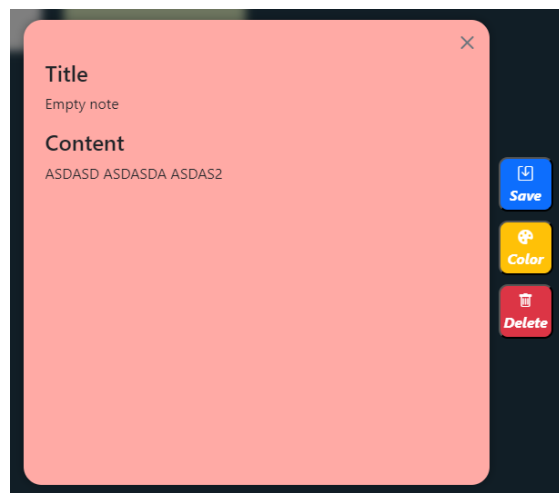
Las notas que cree el usuario se organizan en un **“tablero virtual”**, donde posteriormente se pueden editar o eliminar al pulsar sobre ellas. ¡En un simple vistazo el usuario puede leer de forma rápida todas las notas que tenga guardadas, asegurándose de que no se le olvide nada!



Si el usuario quiere crear una nueva nota, solo deberá de pulsar el botón de “+”, de forma que se creará una nota por defecto vacía, la cual el usuario puede escribir sobre ella, o hacer doble click para editar más propiedades, como el título o el color de fondo de la nota.

Al hacer doble click sobre cualquier nota aparecerá una nueva ventana, en la que se mostrará el contenido de la nota de forma más grande y estructurada. Desde esta nueva ventana, el usuario puede modificar el contenido o cuerpo de la nota, y también puede agregar y modificar el título de la nota, que no es visible desde el tablero principal, pero permite dar contexto y más información a la nota.

Además, en el lateral hay varios botones con diferentes funcionalidades cada uno:



- **Guardar:** Permite al usuario guardar cualquier cambio realizado sobre la nota en esta ventana, ya sea agregar o modificar el título, el cuerpo de la nota o el color, el cual se manejará desde el siguiente botón.
- **Color:** Permite al usuario establecer el color de fondo de la nota. El usuario puede elegir uno de los colores disponibles y establecerlo como color de fondo para la nota. Este color de fondo se verá reflejado tanto en la ventana de edición, como en la vista principal del tablero de todas las notas. El uso de colores puede ayudar al usuario a estructurar mejor las notas, estableciendo un color como una “categoría”, o simplemente puede elegir el color que más le guste visualmente.
- **Borrar:** Permite al usuario borrar la nota del tablero en caso de no necesitarla más.

Las notas se pueden modificar sencilla y de cambiarán los datos de forma automática en la base de datos, el usuario será notificado en la parte inferior derecha con un mensaje de modificación o de creación.

Explicación técnica

El tablero donde se muestran todas las notas realmente no es más que un **<div>** con diferentes **<textarea>** pintados de colores que muestran la información de todas las notas del usuario.

Para ello en el tablero se pinta el botón para añadir más notas, el cual llamaría a la API para indicarle que quiere crear una nueva nota con todos los datos correspondientes.

El método al que llama para añadir la nota simplemente llama a la API, y si la respuesta es correcta crea un nuevo textarea con la información

```
async addNote() {
  let response = await this.addNoteStore();
  !this.notes ? (this.notes = []) : '';
  this.notes.push(response);
},
```

El código que pinta todo el panel de notas es el siguiente:

```
<div id="note-board">
  <div class="add-note text-center" type="button" @click="addNote"></div>
  <textarea v-for="note in notes" :key="note.note_id" spellcheck="false" class="note"
    :style="{ backgroundColor: note.color }" :value="note.content" placeholder="Empty Sticky Note"
    @change="updateNote(note, $event.target.value)" @dblclick="showNote(note)"> </textarea>
  <div v-if="showDialog" class="backdrop">
    <div class="dialog-wrapper">
      <teleport to="body">
        <NoteShow @closeDialog="handleCloseDialog" :note="currentNote" />
      </teleport>
    </div>
  </div>
</div>
```

Se puede observar que realmente se recorre todo el **array** de notas, el cual viene desde el método GET de la API de notas explica posteriormente en el punto 5, relacionado con la API.

Además, a cada nota se le asignan dos eventos de Vue:

- **@change:** Este evento se ejecuta cada vez que algún elemento del **textarea** cambie. Esto es muy útil, pues permite al usuario modificar el texto de las notas directamente desde el panel de forma muy sencilla, sin tener que acceder a todos los datos de la nota para modificarlo. El método al que llama simplemente ejecuta una llamada a la API, la cual se encarga de actualizar el contenido de la nota basándose en el ID de la nota. De esta forma, cada vez que el usuario modifique el contenido de una nota y aparte el ratón de esa nota, se activará el evento **change** y se guardarán los datos en la base de datos, sin que el usuario tenga que molestarse en guardar cada vez.

```
async updateNote(note, newContent) {  
  note.content = newContent;  
  let response = await this.editNoteStore(note);  
},
```

- **@dblclick:** Este evento se ejecuta cada vez que el usuario pulsa dos veces seguidas sobre cualquier nota. Al pulsar dos veces se activa el evento y se cambia una variable **boolean** que indica cuando mostrar y cuando no mostrar la pantalla de modificación de la nota.

Para ello, se hace uso de un componente de Vue. Este componente permanece oculto siempre, excepto cuando el usuario hace doble click. Una vez carga el componente y se muestra la ventana de modificación, el usuario puede no solo modificar el contenido de la nota, sino también el color de fondo o directamente eliminar la nota del tablero.

```
<div v-if="showDialog" class="backdrop">  
  <div class="dialog-wrapper">  
    <teleport to="body">  
      <NoteShow @closeDialog="handleCloseDialog" :note="currentNote" />  
    </teleport>  
  </div>  
</div>
```

4.4 Buylist (Listas de la compra)

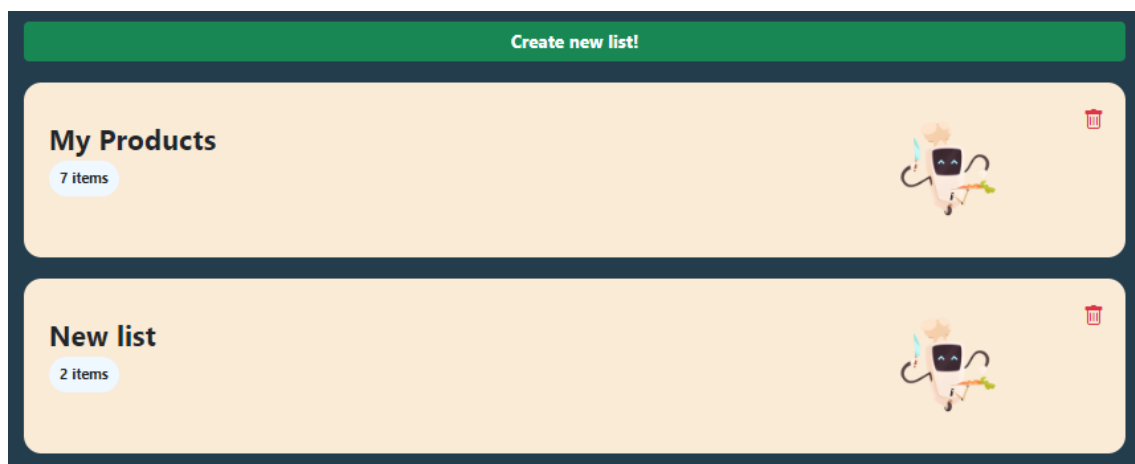
Todo el mundo hace la compra por lo menos una vez a la semana, y muchas veces la compra es tan grande, que nos olvidamos de algunos productos. Algunos recurren a usar una hoja de papel para apuntar las cosas que necesitan, pero este método solo es efectivo un par de veces, pues seguramente el papel se estropee o se pierda, por lo que es necesario hacer la lista de nuevo...

Por ello Mee ha pensado que sería mejor idea tener un lugar donde guardar no solo una si no varias listas al mismo tiempo, y que todas las listas puedan ser modificadas y actualizadas, y lo más importante, ¡que se puedan reutilizar millones de veces!

Explicación general

En la pantalla principal se distinguen dos elementos principales:

- **Botón de crear nuevas listas:** El botón permite al usuario crear una nueva lista vacía. Al pulsar el botón se crea una lista con un nombre por defecto, pero el usuario puede hacer click en el nombre y modificar dicho nombre tantas veces como desee.
- **“Lista de listas”:** Debajo del botón se irán creando las diferentes listas, el usuario podrá ver cuántos elementos contiene la lista en su interior sin necesidad de entrar, pero si quiere ver más detalles deberá de pulsar sobre una de ellas.



Las listas son sencillas y permiten al usuario organizar sus diferentes listas de la compra. Por ejemplo, el usuario puede tener una lista dedicada a productos alimenticios, otra para productos de higiene, o bien puede tener una lista con el nombre del supermercado al que acude frecuentemente, donde puede escribir todo tipo de productos que debe comprar en ese supermercado en específico.

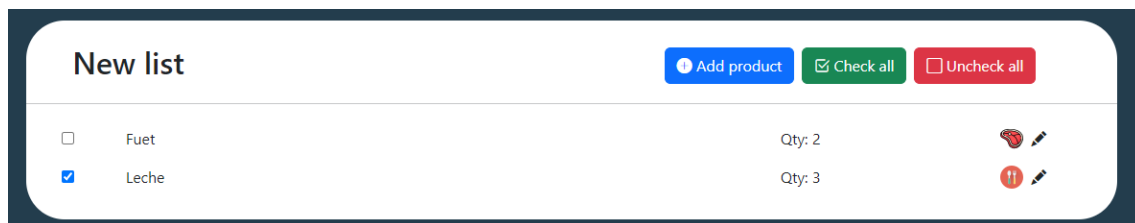
Si el usuario pulsa sobre una de las listas, se mostrarán los detalles concretos de esa lista. Desde esta nueva interfaz, el usuario puede ver todos los productos que hay en la lista.

En esta vista de detalles se pueden observar y marcar como comprados los diferentes productos. Cada producto ocupa una línea de la lista de la compra, y cada uno tiene una casilla de marcado, el nombre del producto, seguido de la cantidad indicada por el usuario y dos iconos:

- **Categoría:** El primer icono hace referencia a la categoría a la que pertenece. Esta categoría es elegida por el usuario durante la creación o modificación de un producto. No afecta en nada, es solo una ayuda visual y de organización para el usuario.
- **Lápiz (editar):** Permite editar los datos del producto seleccionado.

En la parte superior de la lista, hay 3 botones con funcionalidades distintas:

- **Añadir Producto:** Al pulsar, despliega un pequeño formulario donde el usuario puede introducir los datos básicos, como el nombre del producto, la cantidad que quiere comprar, y la categoría. Hay unas pocas categorías por defecto, como productos alimenticios, productos de higiene, lácteos, o bien una categoría general para aquellos productos que no estén claros, o no sea necesario establecer categoría.
- **Marcar (“check”) todo:** Permite marcar todos los productos como “comprados”. Esta función puede ser útil si la lista es corta y el usuario ha comprado todos los productos de golpe, puede marcarlos todos de golpe de forma rápida.
- **Desmarcar (“uncheck”) todo:** Permite desmarcar todos los productos de la lista. Está función es la más útil, pues una vez se haya completado toda la lista, se puede pulsar este botón para hacer un “reset” a la lista y volver a utilizar la misma lista una y otra vez.



Si el usuario compra un producto, puede marcar la casilla de ese producto como comprado, de forma que la lista se va completando poco a poco.

Por último, si se pulsa sobre el botón de editar (icono de lápiz), se puede modificar los datos del producto, como el nombre, la cantidad o la categoría. Además, desde ahí también se permite borrar el producto de la lista.

Explicación técnica

La pantalla principal que muestra todas las listas creadas por el usuario es bastante sencilla, sin muchas complejidades técnicas.

El botón de crear una nueva lista llama a la API, la cual creará una nueva lista en la base de datos, y mostrará dicha lista nueva en el panel.

```
async handleCreateList() {
  let lista = await this.addBuylistStore();
  !this.lists ? (this.lists = []) : '';
  this.lists.push(lista);
},
```

```
<div class="button-wrapper text-center m-4">
  <button class="btn btn-success w-100" @click="handleCreateList">Create new list!</button>
</div>
```

Todas las listas se obtienen a través de la API. Se leen en la base de datos y se muestran al usuario, indicando el nombre y el número de ítems o productos que guarda esa lista.

```
<div class="buycard p-4 m-4" @click="handleShowList(item.buylist_id)" v-for="item in lists" v-if="lists.length > 0">
  <div class="row">
    <div class="col-7 p-3">
      <div class="list-name contenteditable=true spellcheck=false" @click.stop @blur="handleChangeTitle(item, $event.target.textContent)">{{ item.name }}
      <div class="item-chip p-2 text-center">
        <p>{{ item.count }} items</p>
      </div>
    </div>
    <div class="col-4">
      
    </div>
    <div class="col-1">
      <span><i class="bi bi-trash" @click="handleDeleteList(item)" @click.stop style="color: #dc3545; font-size: 24px; float: right;"></i></span>
    </div>
  </div>
</div>
```

Cuando el usuario hace click sobre cualquier lista, se carga el componente que permite editar y crear los productos de esa lista de la compra en concreto.

A la hora de añadir o editar un nuevo producto de la lista, funcionan de forma similar, solo que el añadir envía una petición POST a la API, mientras que el editar envía una petición PUT.

```
saveData(prod) {
  this.showDialog = false;
  this.products.products.push(prod);
  this.currentList[0].elements = JSON.stringify(this.products);
  this.currentList[0].count = this.products.products.length;
  this.editBuylistStore(this.currentList[0]);
},
```

La forma en la que se almacenan los productos es algo distinta a la forma convencional, pues lo normal es tener una tabla de la base de datos donde se guarden las listas, y otra tabla donde se guardan los productos, pero para mantener la integridad de los datos con la versión de Mee Android, decidí mantener el sistema de guardado de productos que usaba la app móvil, que consiste en almacenar la lista de productos como un JSON en un campo de texto en la base de datos. Es por ello que, en la imagen anterior, cuando voy a enviar un producto a la API, convierto el array de productos en formato JSON y se envía a API.

Por último, un detalle importante, se hace uso de una librería de Vue llamada **“debounce”**, creada por Lodash. Esta librería permite añadir cierto retraso a algunas acciones del usuario, como por ejemplo marcar y desmarcar un producto como comprado muchas veces en el mismo segundo, en este caso se añaden unos milisegundos de retardo entre el click y la llamada a la API

```
checkOrUncheck: debounce(function (prod) {
  prod.bought = !prod.bought;
  const itemIndex = this.products.products.findIndex((item) => item.name === prod.name);
  this.currentList[0].elements = JSON.stringify(this.products);
  this.currentList[0].count = this.products.products.length;
  this.editBuylistStore(this.currentList[0]);
}, 100),
```

4.5 Map Gallery (Galería de Mapas)

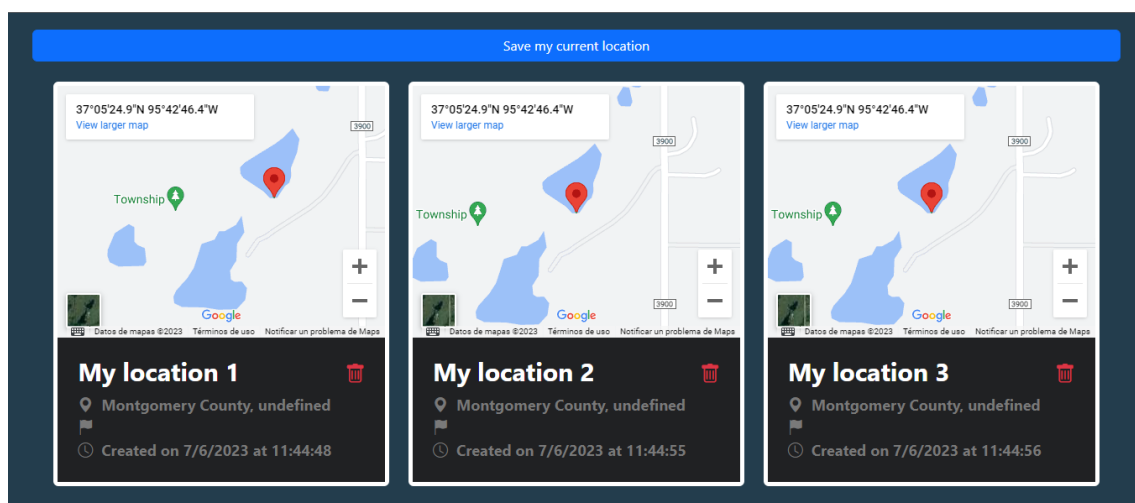
Muchas veces visitamos lugares del mundo increíbles y que todo el mundo conoce, como grandes ciudades, monumentos, museos... Pero no todos los lugares son tan reconocibles y memorables, muchas veces las mejores aventuras ocurren en los lugares más remotos y perdidos. O simplemente puede que nuestra memoria no siempre sea perfecta y olvidemos lugares a los que acudimos periódicamente, o puede que también se nos olvide donde habíamos aparcado nuestro vehículo para ir comprar...

Es por ello por lo que Mee ha creado una galería de ubicaciones, donde el usuario puede guardar la ubicación en la que se encuentra pulsando un solo botón. Más tarde puede volver a consultar esa ubicación en el mapa, e incluso establecer un nombre para preservar ese lugar.

Explicación general

La interfaz es muy sencilla y solo cuenta con dos elementos principales:

- **Botón de guardar ubicación actual:** Con un simple click, y siempre y cuando el usuario otorgue permisos de ubicación, este guardará la posición en la que se encuentra el usuario, extrayendo información relevante como el país y localidad en la que se encuentra. Por defecto, establecerá un nombre de ejemplo, pero el usuario puede modificarlo y establecer el que él quiera.
- **Galería de ubicaciones:** Debajo del botón se muestra la lista de todas las ubicaciones que tiene almacenadas el usuario. Desde cada elemento, se puede consultar el mapa o modificar el nombre, e incluso borrar la ubicación.



Las ubicaciones se quedan almacenadas utilizando Google Maps, de forma que el usuario puede pulsar sobre la tarjeta que contiene la ubicación que quiera consultar y tendrá la opción de abrir esa ubicación en la aplicación móvil o web de Google Maps, desde donde podrá consultar detalles adicionales, o bien establecer la ruta más corta para llegar.

Esta función permite guardar ubicaciones de cualquier lugar del mundo siempre que haya cobertura y permitirá volver a visitar esos lugares en el futuro sin necesidad de tener que buscarlos en un mapa.

Otros casos prácticos podrían ser para acordarse del lugar donde se ha aparcado el vehículo del usuario, pues si visita una ciudad o lugar que no conoce y aparcar lejos, puede que luego no recuerde con exactitud dónde se encuentra el vehículo. De esta forma se puede guardar la ubicación al salir del vehículo y volver a consultarla un rato más tarde.

Explicación técnica

A simple vista, la interfaz de la galería de mapas es bastante sencilla, pues consta de un botón para guardar nuestra ubicación y de una lista o galería de ubicaciones.

Sin embargo, es un poco más complejo de lo que parece, pues se necesitan 2 API's diferentes para obtener todos los datos necesarios de la ubicación del usuario.

Para ello, cuando el usuario pulsa el botón de guarda su ubicación actual, el navegador se asegura de tener permisos de geolocalización, o en caso contrario avisa al usuario si quiere otorgar los permisos o no.

Se ejecuta una llamada a “**navigator.geolocation.getCurrentPosition**”. Esto devolverá un objeto “**position**”, el cual contendrá dos valores, la latitud y la longitud del lugar donde se encuentra el usuario. Usando estos dos valores podemos triangular la posición del usuario haciendo uso de aplicaciones como Google Maps.

```
initGeolocation() {  
  if (navigator.geolocation) {  
    navigator.geolocation.getCurrentPosition(this.success, this.fail);  
  } else {  
    alert('Sorry, your browser does not support geolocation services.');  }  
},  
success(position) {
```

Esos datos se pasan a una nueva función, la cual hará uso de una API llamada **Nominatim Open OpenStreetMap**. Esta API permite obtener datos sobre el lugar que se le pase como coordenadas (es decir, latitud y longitud). En este caso, se pueden obtener datos de diversos niveles, desde el país, nombre de pueblo, ciudad, provincia, hasta incluso el nombre exacto del lugar, o la calle, o el nombre del monumento si de ese caso se tratase, etc.

Pero Mee solo obtiene dos de esos datos, que son el nombre del país y de la provincia o ciudad al que pertenece.

```

success(position) {
  let url = 'https://nominatim.openstreetmap.org/reverse.php?lat=' + position.coords.latitude + '&lon=' + position.coords.longitude + '&format=json';
  axios
    .get(url)
    .then(response => {
      let data = response.data;
      let names = data.display_name.split(',');
      let address = names[0] + ', ' + names[3];
      let created = 'Created on ' + new Date().toLocaleDateString() + ' at ' + new Date().toLocaleTimeString();
      let location = { title: 'Name your location...', address: address, latitude: position.coords.latitude, longitude: position.coords.longitude };
      !this.locations ? (this.locations = []) : '';
      this.locations.push(location);
      this.addMapStore(location);
    })
    .catch(error => {
      console.error(error);
    });
},
fail() {
  console.log('Failed');
},
}

```

Todos los datos obtenidos por la API se procesan y se guarda todo como un objeto JSON, el cual será enviado a la base de datos para guardar la ubicación del usuario para que pueda acceder a ella más tarde.

Además, el objeto JSON que se envía a la base de datos también se muestra en la galería, para que el usuario reciba el feedback de que su ubicación ha sido creada correctamente. Para ello, se hace uso de la API de **Google Maps**, pasándole las coordenadas anteriores como parámetros a un elemento **<iframe>** de HTML, podemos mostrar un pequeño mapa semi-interactivo de la ubicación elegida.

```

<div class="outer-wrapper">
  <button class="btn btn-primary w-100" @click="initGeolocation">Save my current location</button>
  <div class="card-container m-4 v-if="locations.length > 0">
    <div class="card v-for="item in locations" :key="item.id">
      <iframe class="frame-map" :src="https://maps.google.com/maps?q=' + item.latitude + ', ' + item.longitude + '&hl=es'" />
      <div class="card-data p-4">
        <div class="row">
          <div class="col-10">
            <div id="loc-title" class="location-title" contenteditable="true" spellcheck="false" @blur="handleTitleBlur">
              {{ item.title }}
            </div>
            <div class="col-2">
              <span><i class="bi bi-trash" @click="handleDeleteMap(item)" style="color: #dc3545; font-size: 1.2em;"></i></span>
            </div>
          </div>
          <div class="text-span"><i class="bi bi-geo-alt-fill icon-text"></i>{{ item.address }}</div>
          <div class="text-span"><i class="bi bi-flag-fill icon-text"></i>{{ item.info }}</div>
          <div class="text-span"><i class="bi bi-clock icon-text"></i>{{ item.createdAt }}</div>
        </div>
      </div>
    </div>
  </div>
</div>

```

En la imagen de arriba se puede observar la estructura HTML de cada una de las tarjetas, donde se encuentra el botón de guardar ubicación, así como el **<iframe>** con el mapa de Google Maps y la tarjeta inferior con todos los datos (nombre, país, hora de creación, etc).

4.6 Weather (El Tiempo)

Es importante revisar el tiempo que va a hacer antes de salir, es por eso por lo que los humanos siempre tenemos alguna medida para obtener información meteorológica. Mee no quiere quedarse atrás y nos ofrece una funcionalidad para consultar el tiempo actual y de las próximas horas y días gracias a la API de **Weather API**.

Desde esta funcionalidad se puede consultar tanto el pronóstico meteorológico del lugar actual, como de cualquier lugar, pueblo o ciudad del mundo.

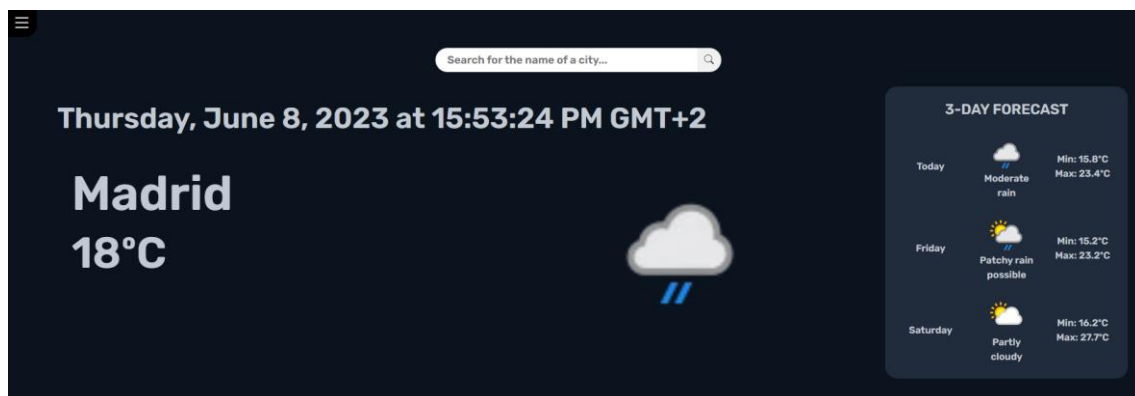
Explicación general

La información se obtiene a través de una llamada a un servicio API REST, es decir, un servidor externo a la aplicación que recolecta los datos y los expone públicamente para que otras aplicaciones, como Mee, puedan hacer uso de ellos. En este caso, y como se menciona anteriormente, se utiliza la API REST de Weather API, la cual proporciona información detallada y ordenada del pronóstico meteorológico de cualquier lugar del mundo.

La interfaz está dividida en varios apartados o tarjetas, donde se muestra una información distinta, pero relacionada con el pronóstico meteorológico del lugar.

La primera tarjeta muestra información general del tiempo en el día y hora actual. Se muestra la fecha y hora de ese mismo momento, y una imagen que simboliza el tiempo actual (soleado, nublado, de noche, noche nublada, etc.).

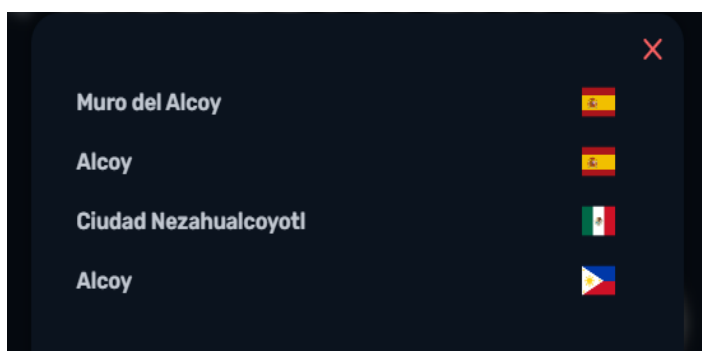
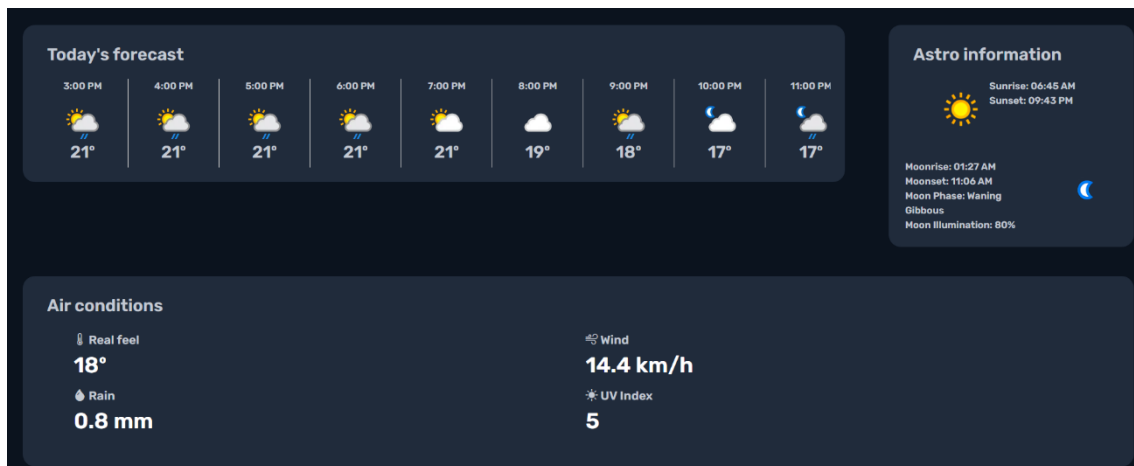
A la derecha se muestra la previsión del tiempo para los siguientes 3 días, donde se muestra el día de la semana, la temperatura media en grados centígrados que hará durante la mayor parte del día, tanto la mínima como la máxima, y una pequeña descripción e imagen del tiempo que hará.



Luego se muestra un historial de temperaturas de todas las horas del día, empezando desde la hora actual, hasta las 00:00 de la noche.

Luego se muestra información relacionada con los astros, el Sol y la Luna, como por ejemplo la hora a la que sale y se esconde el sol (salida y puesta de Sol), y de la misma forma para la Luna, incluyendo también el grado de iluminación y el nombre de la fase actual.

Por último, se muestra información sobre elementos relevantes del pronóstico, como rachas de viento, índice solar, temperatura real, o cantidad de lluvia en las últimas horas.



En la parte superior hay una barra de búsqueda que el usuario puede usar para buscar el tiempo que hace en cualquier parte del mundo. Para ello, debe buscar una cadena de texto con el nombre completo o parte del nombre del lugar que quiere buscar.

Si se encuentra una coincidencia, se mostrará una ventana con todos los nombres que se han encontrado, y el usuario podrá pinchar sobre cualquiera de ellos, haciendo que los datos de la página cambien de acuerdo con el lugar que ha elegido.

Explicación técnica

Todas las diferentes partes que conforman la funcionalidad del tiempo están respaldadas por WeatherAPI.com. Este servicio ofrece un plan gratuito, el cual es muy bueno y provee de mucha información útil sin necesidad de pagar nada.



Ofrece datos en tiempo real sobre el tiempo actual, pronóstico de 3 días y la posibilidad de buscar por país, región o ciudad. No todo es perfecto, pues también tiene sus fallas y limitaciones, por ejemplo, solo trae imágenes 128x128px, las cuales quedan bastante mal si se intentan agrandar o modificar, pero para este proyecto, su plan gratuito es muchísimo más que suficiente.

El primer paso que realiza es obtener un listado de todas (o la gran mayoría) de ciudades o pueblos del mundo. Para ello hace uso de un archivo JSON alojado en **GitHub** por un usuario llamado **"lutangar"**.

De ese archivo se leen todos los registros y se guardan la información importante en un array de lugares o ciudades:

```

getAllCities() {
  fetch('https://raw.githubusercontent.com/lutangan/cities.json/master/cities.json')
    .then((response) => response.json())
    .then((data) => {
      this.citiesArray = data.map((city) => ({
        name: city.name,
        country: city.country,
        latitude: city.lat,
        longitude: city.lng,
      }));
    })
    .catch((error) => console.error(error));
},

```

De cada elemento del fichero JSON, se guardan el nombre del lugar, el país al que pertenece, la latitud y la longitud (las coordenadas para luego enviarlas a la API del tiempo).

Utilizando el array de ciudades, cuando el usuario busque un nombre en la barra de búsqueda, si se encuentra alguna ciudad o lugar que contiene la cadena que ha escrito el usuario, se muestra el nombre del lugar y la bandera del país al que pertenece. Para la bandera se utiliza una librería externa que permite mostrar diferentes banderas de países en diferentes tamaños. Esta librería se conoce como “**vue-country-flag-next**”.

```

<div class="location-row my-0" v-for="item in locations" :key="item.name" @click="selectLocation(item)">
  <div class="row">
    <div class="col-10 px-4">
      {{ item.name }}
    </div>
    <div class="col-2">
      <country-flag :country="item.country" size="normal"></country-flag>
    </div>
  </div>
</div>

```

La parte principal de esta funcionalidad es el método que se encarga de obtener los datos de **WeatherAPI**. Todo el resto de partes funciona en base a los datos que se obtienen de este método.

```

getTodayForecast(lat, lng) {
  axios
    .get('https://api.weatherapi.com/v1/forecast.json?key=_____&q=' + lat + ',' + lng + '&days=7&aqi=yes&alerts=no%22')
    .then((response) => {
      this.forecast = response.data.forecast.forecastday[0];
      this.currentWeather = response.data.current;
      this.forecast3days = response.data.forecast.forecastday;
      this.filteredHours = [];
      const now = new Date().getHours();
      this.forecast.hour.forEach((e) => {
        if (new Date(e.time) > new Date() || new Date(e.time).getHours() == now) {
          this.filteredHours.push(e);
        }
      });
    })
    .catch((err) => alert(err));
},

```

El método recibe como parámetros la latitud y la longitud (por defecto apuntan a Madrid, pero si el usuario utiliza el buscador de ciudades, este método recibirá las coordenadas del lugar que elija el usuario).

Realiza la llamada a WeatherAPI y toda la información que se recibe se almacena en tres partes o arrays diferentes:

- **Forecast:** Almacena el pronóstico del tiempo durante todo el día, es decir, no solo muestra la información relevante, sino que también tiene un array desde las 00:00 hasta las 23:00 con la información relevante del pronóstico de tiempo que hará cada hora del día.
- **Current Weather:** Almacena el pronóstico del tiempo en ese mismo momento. Además, es el que contiene datos adicionales relevantes como índice solar, velocidad del viento, porcentaje de lluvia, presión atmosférica, entre otros muchos.
- **Forecast 3 Days:** Almacena el pronóstico del tiempo para los próximos 3 días. Es muy parecido a currentWeather, pero esta vez para 3 días en vez de solo hoy.

Utilizando los datos de esos 3 arrays, se monta el HTML con todos los datos, cada uno en el lugar que le corresponde.

4.7 News (Noticias)

Todas las personas tienen curiosidad por ver que está ocurriendo a su alrededor, ya sea a una distancia cercana, o bien en la otra parte del mundo. Hoy en día tenemos gente que se dedica a obtener información sobre los acontecimientos importantes del mundo y nos los presentan en forma de noticias, ya sea en un canal de noticias, foros, blogs de internet, periódicos de papel o digitales, etc.

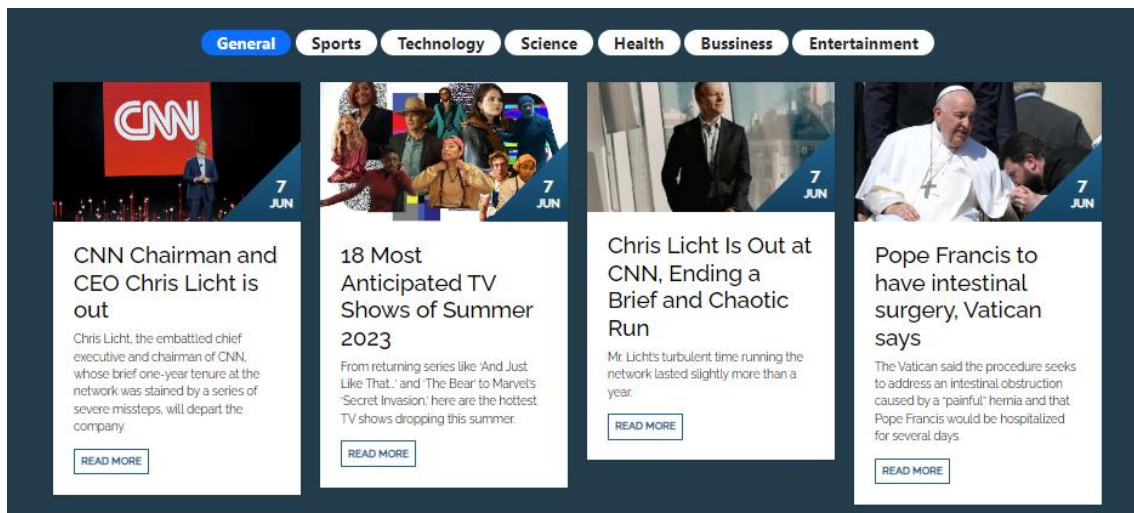
Es por eso por lo que Mee ha pensado que sería una buena idea que el usuario tenga acceso a varias noticias desde un lugar de fácil acceso. Gracias a esta funcionalidad, el usuario podrá consultar noticias de diferentes categorías sin tener que realizar ninguna búsqueda.

Explicación general

Las noticias que se muestran en la pantalla son noticias obtenidas a través de un servicio de API Rest, desde el cual se lee una lista de noticias y se muestran al usuario en un formato más amigable.

La interfaz de noticias es muy sencilla. Cuando el usuario accede a esta funcionalidad, por defecto se cargarán noticias de carácter general, es decir, se mostrarán todas las noticias más importantes que devuelva la API, sin importar la categoría a la que pertenecen.

Sin embargo, el usuario dispone de una serie de botones en la parte superior que permiten elegir el tipo de noticias que quiere consultar por categorías. Las categorías más relevantes y que puede consultar son: Generales, Deportes, Tecnología, Ciencia, Salud, Negocios y Entretenimiento.



Por cada noticia que trae la API, el usuario recibe la siguiente información relevante de la noticia:

- **Imagen:** Lo primero que ve y que más resalta es la imagen de la noticia, la cual permite dar contexto adicional y entender mejor el propósito de la noticia.
- **Titular:** La parte más importante de la noticia, el cual permite entender en pocos segundos de lo que se trata la noticia y sus características.
- **Descripción:** Además del titular, se incluye una breve descripción donde se entra más en detalle sobre el tema del que trata la noticia.
- **Enlace original:** Si el usuario pulsa el botón de leer más, se le redirigirá a la fuente oficial donde se ha publicado la noticia, donde seguramente encuentre toda la información relevante de la noticia.

Explicación técnica

Nota: Durante el desarrollo de la aplicación Mee Web, la API que utilicé para extraer noticias en la aplicación Android de Mee, News API, cambió sus políticas de uso en la versión gratuita y deshabilitaron la posibilidad de realizar peticiones desde fuera de localhost, por lo que, al publicar la web, las noticias nunca aparecerían. Por ello me vi obligado a buscar una alternativa, siendo en este caso GNewsAPI. El único problema que tuve con esta API era que la cantidad de noticias por petición era mucho menor, pero al menos mantenía casi toda la estructura JSON, por lo que adaptar el código no fue demasiado difícil.

La interfaz se divide en dos partes, la barra de selección de categorías y la lista de noticias sobre esa categoría.

Para cambiar de categoría, el usuario solo debe hacer click sobre cualquiera de los nombres de la categoría, lo que invocará un método que llamará al método principal que obtiene todas las noticias de la API, así como cambiar el botón de categoría activo.

```

changeCategory(cat) {
  this.loadNews(cat);
  if (document.getElementsByClassName('bg-primary')[0]) {
    document.getElementsByClassName('bg-primary')[0].classList.remove('bg-primary');
    document.getElementsByClassName('text-white')[0].classList.remove('text-white');
  }
  event.currentTarget.classList.add('bg-primary');
  event.currentTarget.classList.add('text-white');
},

```

El método principal es **loadNews**. Este realiza la llamada a la API de **GNews**, filtrando por la categoría que ha elegido el usuario (o por general por defecto). Todas las noticias que obtiene de la API se guardan en un array y se muestran luego al usuario.

```

loadNews(category) {
  this.news = [];
  let url = 'https://gnews.io/api/v4/top-headlines?lang=en&country=us&max=10';
  if (category) {
    url = url + '&category=' + category;
  } else {
    url = url + '&category=general';
  }
  axios
    .get(url + '&apikey=9116e08db9f993e7094071a2c1da334b')
    .then((response) => {
      let notic = response.data.articles;
      this.news = notic;
      notic.forEach((e) => this.news.push(e));
    })
    .catch((err) => alert(err));
},

```

Luego el array de noticias se recorre en la parte HTML, y se muestran los datos que contiene cada uno de las noticias del array.

```

<div class="col-lg-3 col-md-6 col-12 mx-auto news-feed" v-for="item in news" :key="news.id">
  <figure class="snip1237">
    <div class="image">
      <i class="ion-ios-clock-outline"></i>
      <div class="date">
        <span class="day">{{ calculateDay(item.publishedAt) }}</span>
        <span class="month">{{ calculateMonth(item.publishedAt) }}</span>
      </div>
    </div>
    <figcaption>
      <h3>{{ item.title }}</h3>
      <p>{{ item.description }}</p>
      <a @click="showNews(item.url)" class="read-more">Read More</a>
    </figcaption>
  </figure>
</div>

```

Por cada noticia, se muestra la imagen que viene adjunta (o una por defecto en caso de no tener imagen), la fecha de publicación de dicha noticia, el título y descripción breve de la noticia y un botón que contiene el enlace a la noticia original en la página que ha sido publicada, en caso de que el usuario necesite investigar más al respecto.

4.8 Words (Palabras)

Muchas veces necesitamos acceso a un diccionario o a un traductor. Pero nunca sabemos cuál usar, o bien no nos acordamos del nombre del que más nos gusta.

Es por eso por lo que Mee a preparado una función especial que ha bautizado como “Palabras”. En esta multiherramienta el usuario tiene acceso rápido tanto a un traductor de muchos idiomas y bidireccional, como a un diccionario con varias definiciones.

Además, como regalo, Mee dejará cada día una palabra nueva en la parte de detrás de una tarjeta, ¡por lo que el usuario puede hacer girar la tarjeta para descubrir una nueva palabra cada día y añadirla a su vocabulario!

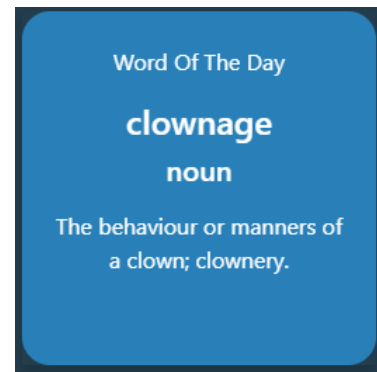
En esta funcionalidad se usan 3 API's distintas para obtener los servicios necesarios, en este caso para el diccionario, para el traductor y para la palabra del día.

La interfaz es muy simple. Consta de 3 “tarjetas” diferentes, cada uno con su icono y nombre correspondiente. Las tarjetas tienen las siguientes funcionalidades al pulsarlas:

4.8.1 Word of The Day (Palabra del día)

Explicación general

Cada día se buscará una nueva palabra con su significado. Si el usuario pulsa sobre esta tarjeta, esta se dará la vuelta y mostrará la palabra en cuestión, junto con su definición. De esta forma el usuario aprenderá de forma interactiva una nueva palabra cada día.



Explicación técnica

Para obtener la palabra cada día se utiliza una Api externa llamada **WordnikAPI**. Esta es una API gratuita relacionada con el mundo de las palabras. Entre todas las funciones que tienen online, también tiene una API que devuelve una palabra aleatoria cada día.



Suelen ser palabras extrañas o poco comunes, lo que la hace perfecta para que el usuario amplíe su vocabulario.

La llamada a la API devuelve el siguiente objeto JSON, el cual contiene los datos que se muestran más tarde, como la palabra en sí, la definición, si se trata de un verbo, un adjetivo, etc., e incluso ejemplos de uso.

```
{
  "_id": "647610b5114ce323da8a1345",
  "word": "resistentialism",
  "publishDate": "2023-06-08T02:00:00.000Z",
  "contentProvider": { ... }, // 2 items
  "note": "This word is a blend of 'resist' and 'existentialism.'",
  "htmlExtra": null,
  "pdd": "2023-06-08",
  "definitions": [
    {
      "text": "The theory that certain inanimate objects hate humans.",
      "partOfSpeech": "noun",
      "source": "wiktionary",
      "note": null
    }
  ],
  "examples": [ ... ] // 3 items
}
```

Esos datos se procesan en el código y se muestran al usuario como una tarjeta HTML interactiva, la cual al ser pulsada se dará la vuelta, como si el reverso de una carta se tratase, y mostrará la palabra y su significado.

Todos los datos se guardan en un array llamado **“wotd (Word of the day)”** y es muestran de la siguiente forma:

```
<div class="flip-card p-4" @click="flipCard">
  <div class="flip-card-inner">
    <div class="flip-card-front p-3">
      <h3>Word Of The Day<br />Tap to reveal!</h3>
      
    </div>
    <div class="flip-card-back p-4">
      <p>Word Of The Day</p>
      <h4 class="word mt-2">{{ wotd.word }}</h4>
      <h5 class="mb-3">{{ wotd.definitions[0].partOfSpeech }}</h5>
      <p>{{ wotd.definitions[0].text }}</p>
    </div>
  </div>
</div>
```

4.8.2 Translator (Traductor)

Explicación general

Al pulsar sobre la tarjeta, se mostrará la interfaz del traductor. Esta es muy sencilla y consta de dos elementos: el texto que escribe el usuario en el idioma que quiere usar y el texto que se generará en el idioma que se indique.

Por defecto, si el usuario no elige ningún idioma de entrada, el traductor intentará adivinar de que idioma se trata y lo traducirá al inglés.

Sin embargo, el usuario puede modificar ambos idiomas de entrada y salida pulsando el desplegable de idiomas en la parte superior. En él se encuentra una lista con todos los idiomas disponibles en el traductor. A pesar de no contener todos los idiomas en el mundo, tiene la gran mayoría de los idiomas más conocidos y hablados alrededor del mundo.

Si se pulsa el botón de traducir, la palabra se enviará para traducir y aparecerá en el campo de texto de la derecha, traducido al idioma indicado.

Si se pulsa el botón de reset, ambos textos en los campos volverán a estar vacíos y se podrá continuar usando el traductor.

Por último, el botón central permite cambiar el orden de traducción, convirtiéndolo en un traductor bidireccional, de forma que, si el usuario estaba escribiendo de español a inglés, ahora podrá traducir de inglés a español.

Explicación técnica

La programación principal en esta funcionalidad es, obviamente, la API encargada de la traducción de los textos o palabras que escriba el usuario.

Para ello, he decidido utilizar una API de **Microsoft Azure**, alojada en **RapidAPI**. Esta API es muy buena, siendo de Microsoft, una empresa muy conocida y avanzada con su tecnología de Inteligencia Artificial, y que en este caso cumple los requisitos necesarios para ser usada como traductor de Mee. Además, ofrece un plan gratuito, lo cual facilita su uso.

EL traductor tiene dos modos de uso, y según su modo de uso, la llamada a la API es algo diferente, pero ambas llamadas apuntan al mismo sitio, cambiando ciertos parámetros dentro de la llamada. Los dos modos de traducción son los siguientes:

- **Traducción A -> B:** En este modo el usuario elige los dos idiomas, tanto el de origen como el de destino. De esta forma el usuario puede especificar exactamente desde el idioma que quiere traducir. Para ello, debe seleccionar los dos idiomas de la lista de idiomas proporcionada por la API en los selectores de idioma.

```
translate() {
  let textFrom = document.getElementById('input-from');
  let textTo = document.getElementById('input-to');
  this.langFrom = document.getElementById('select-from').value;
  this.langTo = document.getElementById('select-to').value;
  const options = {
    method: 'POST',
    url: 'https://microsoft-translator-text.p.rapidapi.com/translate',
    params: {
      'api-version': '3.0',
      'to[0]': this.langTo,
      textType: 'plain',
      profanityAction: 'NoAction',
      from: this.langFrom,
    },
    headers: {
      'content-type': 'application/json',
      'X-RapidAPI-Key': 'abace8d9b9msh4cac9c2f0610797p18ab96jsn0d2a2de2bb2c',
      'X-RapidAPI-Host': 'microsoft-translator-text.p.rapidapi.com',
    },
    data: '[{"Text":"' + textFrom.textContent + '"}]',
  };
  axios
    .request(options)
    .then(function (response) {
      textTo.textContent = response.data[0].translations[0].text;
    })
    .catch(function (error) {
      console.error(error);
    });
}
```

```
detectTranslate() {
  let textFrom = document.getElementById('input-from');
  let textTo = document.getElementById('input-to');
  this.langTo = document.getElementById('select-to').value;
  const options = {
    method: 'POST',
    url: 'https://microsoft-translator-text.p.rapidapi.com/translate',
    params: {
      'api-version': '3.0',
      'to[0]': this.langTo,
      textType: 'plain',
      profanityAction: 'NoAction',
    },
    headers: {
      'content-type': 'application/json',
      'X-RapidAPI-Key': 'abace8d9b9msh4cac9c2f0610797p18ab96jsn0d2a2de2bb2c',
      'X-RapidAPI-Host': 'microsoft-translator-text.p.rapidapi.com',
    },
    data: '[{"Text":"' + textFrom.textContent + '"}]',
  };
  axios
    .request(options)
    .then(function (response) {
      textTo.textContent = response.data[0].translations[0].text;
      document.getElementById('select-from').value = response.data[0].detectedLanguage.language;
    })
    .catch(function (error) {
      console.error(error);
    });
}
```

- **Detectar idioma:** En este modo el usuario solo elige el idioma final, mientras que el idioma origen será detectado automáticamente por la API de Microsoft Azure. Esto permite que el usuario no tenga que elegir el idioma, ya que en algunos casos puede que copie el texto de algún lado y no conozca de que idioma proviene, pero esto también puede originar errores, ya que la API intentará adivinar de que idioma se trata, pero siempre puede equivocarse.

4.8.3 Meectionary (Diccionario)

Explicación general

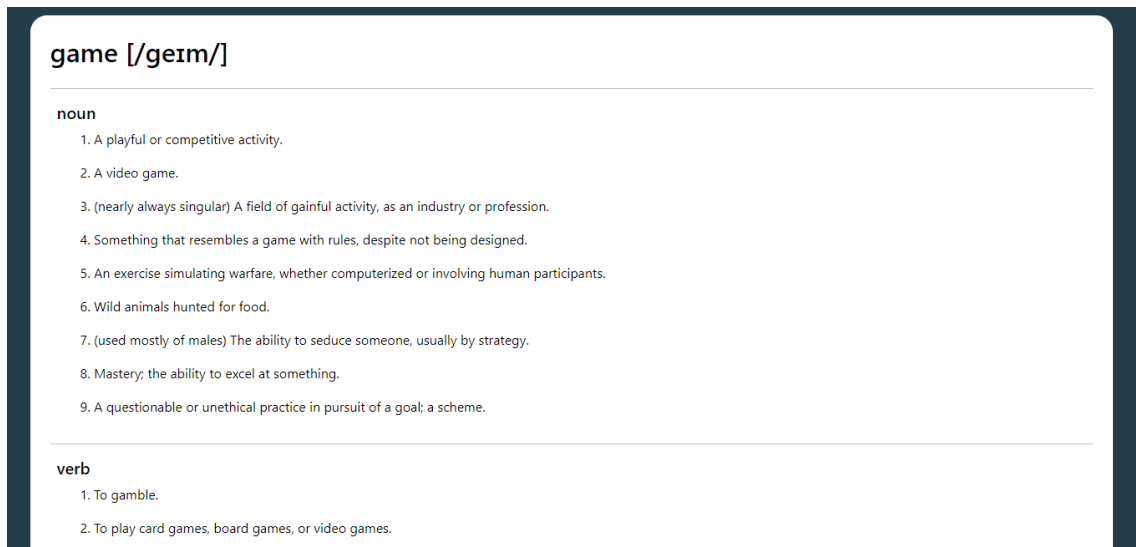
La interfaz es simple, solo hay una barra de búsqueda y un botón de buscar. El usuario puede escribir la palabra de la cual quiere conocer el significado en la barra de búsqueda y pulsar el botón de la lupa para efectuar la búsqueda.



Si la palabra existe en el diccionario inglés, se mostrarán una serie de definiciones en la parte inferior de la barra de búsqueda.

También se muestra la pronunciación fonética de la palabra que ha buscado, y dentro de las definiciones se muestran, no solo las definiciones de la palabra, si no que se ordenan por sustantivo, adjetivo, verbo, etc.

Por cada tipo de los anteriores se muestra una serie de definiciones relevantes para la palabra indicada.



Explicación técnica

Para esta funcionalidad del diccionario, se hace uso de una API llamada DictionaryDev. Esta herramienta de diccionario es muy sencilla y además es gratuita. Si le pasamos una palabra y la encuentra en el diccionario, nos mostrará una serie de definiciones de la palabra, agrupándolas por el tipo de palabra, es decir, todas las definiciones en la que actúa como verbo, todas las definiciones en las que actúa como sustantivo, etc.

La API devuelve una estructura como la siguiente cuando realizamos una petición. En este caso, la palabra elegida es apple (manzana):

```
"word": "apple",
"phonetic": "/ˈæp.əl/",
"phonetics": [...], // 2 items
"meanings": [
  {
    "partOfSpeech": "noun",
    "definitions": [
      {
        "definition": "A common, round fruit produced by the tree Malus domestica, cultivated in temperate climates.",
        "synonyms": [],
        "antonyms": []
      },
      {
        "definition": "Any of various tree-borne fruits or vegetables especially considered as resembling an apple; also",
        "synonyms": [],
        "antonyms": []
      }
    ]
  }
]
```

La llamada a la API es sencilla, pues guardamos la palabra que ha escrito el usuario en una variable global llamada “Word”, y la usamos en la petición a la API para obtener el JSON con todos los datos. Como los datos que interesan son las definiciones, estas se guardan en un array llamado “meanings”:

```
async setWord(userWord) {
  this.word = userWord;
  let response = await axios.get('https://api.dictionaryapi.dev/api/v2/entries/en/' + this.word);
  this.meanings = response.data;
},
```

Utilizando todos los datos que nos devuelve, se monta un componente de Vue con todos esos datos ordenados y mostrados de forma que sea agradable y entendible para el usuario.

A continuación, se muestra el fragmento de código que muestra todas las definiciones, guardadas en un array llamada “meanings”, que contiene todas las definiciones de la misma palabra en sus diferentes tipos (adjetivo, verbo, sustantivo).

```
<div class="container mt-5 pt-5">
  <div class="dictionary p-4" v-if="meanings[0]">
    <div class="row">
      <h2>{{ meanings[0].word }} [{{ meanings[0].phonetic }}]</h2>
    </div>
    <hr class="hr" />
    <h5 class="px-2">{{ meanings[0].meanings[0].partOfSpeech }}</h5>
    <div class="row">
      <p class="px-4 mx-md-4" v-for="(item, index) in meanings[0].meanings[0].definitions">{{ index + 1 }}. {{ item.definition }}</p>
    </div>
    <hr class="hr" />
    <h5 class="px-2">{{ meanings[0].meanings[1].partOfSpeech }}</h5>
    <div class="row">
      <p class="px-4 mx-md-4" v-for="(item, index) in meanings[0].meanings[1].definitions">{{ index + 1 }}. {{ item.definition }}</p>
    </div>
  </div>
</div>
```

5. Base de Datos

En cualquier proyecto de desarrollo, sin importar la dificultad o cantidad de contenido, una base de datos es un componente esencial que almacena, organiza y gestiona de manera estructurada un conjunto de datos relacionados. Es una herramienta que permite acceder, manipular y recuperar información de manera eficiente para ser utilizada en el desarrollo.

El uso de una base de datos en un proyecto como este se justifica por varias razones fundamentales:

- **Organización:** Una base de datos proporciona un medio para organizar los datos de manera estructurada. Los datos se guardan en tablas con filas y columnas, lo que facilita la gestión y la recuperación de información de manera eficiente en cualquier aplicación, ya sea web, móvil u otra.
- **Acceso y manipulación eficientes de los datos:** Una base de datos ofrece mecanismos y lenguajes de consulta (como SQL) que permiten realizar operaciones de búsqueda, actualización, eliminación e inserción de datos de forma rápida y precisa. Con el propio lenguaje SQL se consigue manipular los datos para lo que se han de usar.
- **Integridad y consistencia de los datos:** Al utilizar una base de datos, es posible aplicar restricciones, filtros y reglas de integridad para garantizar que los datos cumplan con determinadas condiciones y reglas. Esto ayuda a mantener la consistencia y la integridad de los datos, evitando errores y garantizando que la información que se almacena en ella sea correcta y adecuada.
- **Escalabilidad y capacidad de gestión de datos:** Una base de datos está diseñada para manejar grandes cantidades de datos de manera eficiente, lo que permite no solo almacenar toda la información de la aplicación web, sino que además permite prever un posible crecimiento futuro de la cantidad de datos almacenados, pudiendo escalar rápidamente para adaptarse a los nuevos datos.
- **Compartir y acceder a los datos de manera simultánea:** Una base de datos permite que múltiples usuarios accedan y utilicen datos de manera simultánea, permitiendo que todos ellos puedan usar la aplicación web sin problemas de integridad ni choque de datos.

En este caso, la base de datos creada para la aplicación web no está perfectamente estructurada o alguna tabla no está planteada óptimamente, como por ejemplo la tabla de listas de la compra almacena todos los productos como un gran campo de texto en formato JSON, cuando realmente debería haber una tabla de listas y una tabla de productos, conectadas entre sí.

La decisión de hacerlo de la otra forma es para preservar la futura compatibilidad entre la aplicación web y la aplicación móvil, ya que en la base de datos del proyecto del año pasado se configuraron las tablas esa forma, y he decidido mantener esa estructura de tablas.

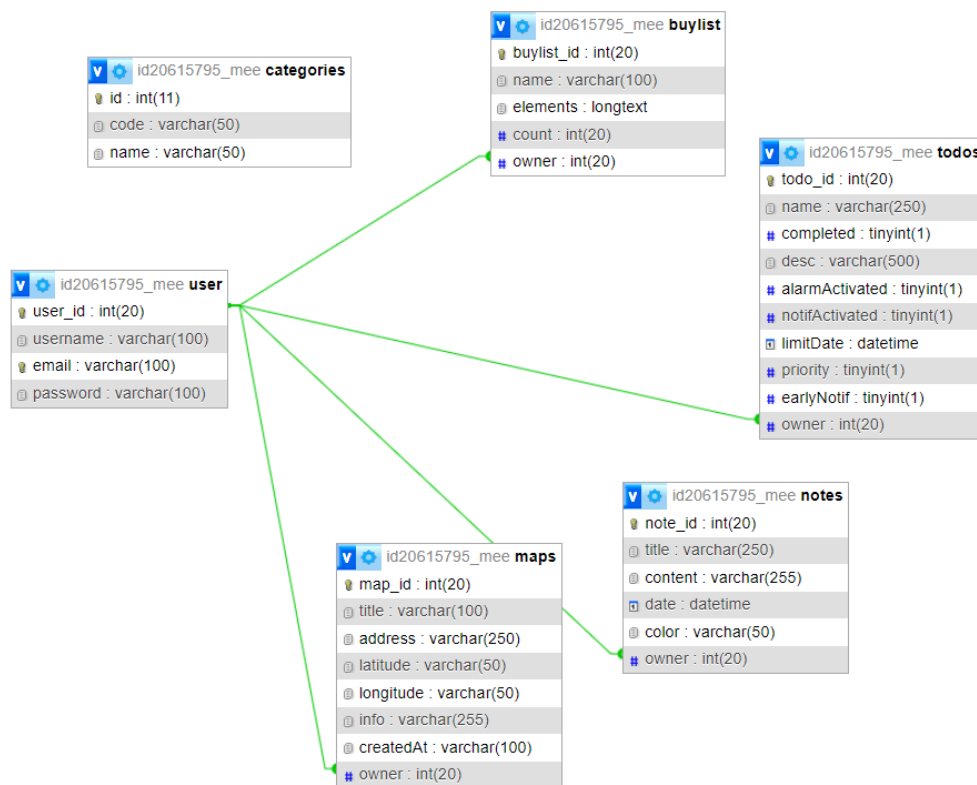
5.1 Esquema ER (Entidad-Relación)

Un esquema Entidad-Relación es un modelo conceptual utilizado en el diseño de bases de datos para representar las entidades, atributos y relaciones entre ellas. Proporciona una forma visual de comprender la estructura y las conexiones en una base de datos, lo que facilita el diseño y la organización de la información.

El ER es el primer paso que realizar en cualquier proyecto, pues este permite crear una idea visual de los diferentes elementos y datos que almacenará la base de datos una vez creada.

Es importante crear el ER justo al principio, ya que nos permite identificar posibles errores o mejoras a la base de datos incluso antes de crearla. Rectificar un error en el esquema es sencillo y cuesta segundos, pero rectificar un error en una base de datos que ya ha sido creada puede ser perjudicial e incluso causar problemas con los datos que ya hay almacenados.

Es por ello por lo que para el desarrollo de Mee también se ha realizado un esquema entidad-relación con todas las tablas y atributos presentes en la base de datos final:



5.2 Estructuración de tablas

A continuación, se explican los usos y utilidades de cada una de las tablas, así como sus campos:

Tabla “user”

Esta es la tabla principal. De esta tabla dependen el resto de las tablas de la base de datos, ya que para organizar los datos y que cada usuario utilice la aplicación solo tenga acceso a sus propios datos, y no a los de nadie más, es necesario que todos los datos tengan un identificador del usuario al que pertenecen.

Esta tabla está compuesta por los siguientes atributos:

- **User_id:** Identificador del usuario. Esta es su clave primaria y que permite que no haya dos usuarios iguales en la misma tabla, incluso si hubiera otro con el mismo nombre. Este atributo es el más importante de toda la base de datos, pues como se ha comentado anteriormente, aparece en todas las tablas como clave ajena, es decir, todas las tablas tienen un campo “owner” que apunta directamente a este “user_id”, de esa forma es fácil identificar y buscar, por ejemplo, todas las notas del usuario con ID 23. También se utiliza como identificador para el actualizado y borrado.
- **Username:** Nombre o alias del usuario que se ha registrado y que se mostrará en el menú desplegable para identificar el usuario con el que se ha conectado.
- **Email:** Atributo que indica la dirección de correo registrada por el usuario. Servirá como identificador para acceder después de que el usuario se registre. Es importante acordarse del correo y la contraseña para poder acceder a la aplicación web Mee.
- **Password:** La contraseña única y totalmente privada de cada usuario. Esta se establece durante el registro y se utilizara junto al mail para acceder a la aplicación a través del login.

Tabla “notes”

En esta tabla se almacenan todas las notas de todos los usuarios.

Esta tabla está compuesta por los siguientes atributos:

- **Note_id:** Identificador único que diferencia inequívocamente cada nota del resto de notas almacenadas.
- **Title:** Almacena el nombre o título de la nota.
- **Content:** Almacena el cuerpo o descripción de la nota. Se actualiza constantemente, cuando el usuario modifica su nota desde el tablero de notas.
- **Date:** Almacena la fecha de creación de la nota, tanto la fecha de creación como la hora de creación.
- **Color:** Almacena el código hexadecimal que hace referencia al color de fondo de la nota elegido por el usuario durante la edición de nota.
- **Owner:** Clave ajena que apunta a “user_id” de la tabla user. Permite identificar que la nota pertenece al usuario X y realizar operaciones de modificación, borrado y búsqueda de notas usando este campo como filtro.

Tabla “todos”

En esta tabla se almacenan todas las tareas de todos los usuarios.

Esta tabla está compuesta por los siguientes atributos:

- **todo_id:** Identificador único que diferencia inequívocamente cada tarea del resto de tareas almacenadas.
- **Name:** Almacena el nombre o título de la tarea.
- **Completed:** Booleano que indica si la tarea esta completada o no
- **Desc:** Almacena el cuerpo o descripción de la nota.
- **AlarmActivated:** Booleano que indica si la notificación por alarma esta activada o no.
- **LimitDate:** Almacena la fecha límite o fecha de vencimiento para completar la tarea.
- **Priority:** Almacena un número que representa la prioridad que ha elegido el usuario.
- **EarlyNotif:** Almacena un número que representa la cantidad de tiempo elegida por el usuario para la notificación adelantada que ha elegido el usuario.
- **Owner:** Clave ajena que apunta a “user_id” de la tabla user. Permite identificar que la tarea pertenece al usuario X y realizar operaciones de modificación, borrado y búsqueda de tareas usando este campo como filtro.

Tabla “maps”

En esta tabla se almacenan todos los mapas de todos los usuarios.

Esta tabla está compuesta por los siguientes atributos:

- **map_id:** Identificador único que diferencia inequívocamente cada mapa del resto de mapas almacenados.
- **Title:** Indica el nombre que le ha puesto el usuario al mapa.
- **Address:** Almacena el nombre del lugar (pueblo o ciudad).
- **Latitude:** Almacena la coordenada latitud del lugar.
- **Longitude:** Almacena la coordenada longitud del lugar.
- **Info:** Almacena la comunidad o país donde se encuentra el lugar.
- **createdAt:** Almacena la fecha de creación.
- **Owner:** Clave ajena que apunta a “user_id” de la tabla user. Permite identificar que el mapa pertenece al usuario X y realizar operaciones de modificación, borrado y búsqueda de mapas usando este campo como filtro.

Tabla “buylist”

En esta tabla se almacenan todas las listas de compra de todos los usuarios.

Esta tabla está compuesta por los siguientes atributos:

- **buylist_id:** Identificador único que diferencia inequívocamente cada lista de la compra del resto de mapas almacenados.
- **Name:** Indica el nombre que le ha puesto el usuario a la lista.
- **Elements:** Almacena todos los productos y cantidades de cada producto, así como la categoría y nombre, en formato JSON.
- **Count:** Indica el número total de productos en la lista.
- **Owner:** Clave ajena que apunta a “user_id” de la tabla user. Permite identificar que la lista de la compra pertenece al usuario X y realizar operaciones de modificación, borrado y búsqueda de listas de compra usando este campo como filtro.

Tabla “categories”

En esta tabla se almacenan las categorías disponibles para los productos.

Esta tabla está compuesta por los siguientes atributos:

- **id:** Identificador único que diferencia inequívocamente cada categoría del resto.
- **Name:** Indica el nombre de la categoría.
- **Code:** El código o identificador en formato texto.

5. Application Programming Interface (API)

Una Interfaz del Programa de Aplicación (Application Programming Interface) es un conjunto de reglas que permite que varios programas se comuniquen entre sí y se transfieran datos de uno al otro. Para ello los datos se almacenan en un servidor, el cual permite a varios clientes extraer esos datos a través de llamadas HTTP.

En la mayoría de las aplicaciones web, incluyendo Mee, utilizan las API para comunicarse con el servidor donde se almacena el sistema gestor de bases de datos, de forma que desde el cliente (en este caso el ordenador de un usuario) se envían peticiones HTTP (GET, PUT, POST, DELETE) para obtener, modificar, añadir o eliminar datos de la base de datos sin tener que comunicarse con esta de forma directa.

5.1 Plan inicial

Desde el planteamiento de la idea de crear Mee como una aplicación web, tenía pensado crear tanto la base de datos como la API utilizando un framework de PHP conocido como Laravel. Este framework permite crear una base de datos y una API que ataque a esa misma base de datos de una forma muy sencilla.

Sin embargo, durante el curso ya habíamos trabajado con Laravel, por lo que sería hacer lo mismo que hicimos en clase. Otro motivo por el que descarté hacer un proyecto de Laravel fue el hecho de que el servidor o máquina virtual debería de estar todo el tiempo activo para que funcionar, y buscar un hosting gratuito online para Laravel es más complicado.

Por tanto, tomé la decisión de eliminar opción de usar Laravel y decidí probar algo nuevo, con lo que nunca había trabajado antes. Decidí hostear la API en un servidor Apache-PHP, es decir, un servidor normal y corriente, básico, sin necesidad de ningún tipo de framework que agilice las tareas.

De esta forma podría mantener el servidor de API funcionando a cualquier hora y en un futuro el hosting podría ser escalable, si la aplicación se publicara para todo el mundo.

5.2 Hosting

Es por ello por lo que después de buscar diferentes alternativas de hosting online, me decidí por 000webhost. Este servicio de hosting tiene un plan gratuito muy bueno, y de hecho durante una práctica este curso lo utilizamos para subir una pequeña web.

Tan solo con el plan gratuito, tienes acceso a la raíz del servidor, para subir todos los ficheros de una web, aplicación, api, etc. Además, ofrece acceso a una base de datos gratuita con conexión remota a través de PHPMyAdmin desde donde se puede crear y configurar las tablas de la base de datos, y por último un servicio de FTP con editor online, por lo que crear los ficheros y carpetas PHP ha sido super sencillo utilizando el FTP gráfico que ofrece.



Por supuesto no todo es bueno, pues al ser un plan gratuito no se pueden almacenar demasiados registros de base de datos y ficheros, por lo que una aplicación muy grande sería muy difícil hostearla en un lugar así, pero en mi caso, para hostear Mee-Web tengo recursos más que suficientes.

La latencia también es algo mala, pues el servidor está hosteado en otra parte del mundo y para ejecutar sentencias o cargar datos suele tardar un poco más de lo esperado, pero incluso con todos los contras mencionados, las ventajas que ofrece superan a las desventajas, por lo que finalmente decidí usarlo como hosting principal para el servidor de BD y API de Mee Web.

5.3 Resolución

El planteamiento por el que he optado a la hora de crear la API ha sido el siguiente: En la raíz del servidor hay una carpeta llamada API.

Cada funcionalidad de la aplicación web (e.g notas, galería de mapas, etc.) hay una carpeta creada en el servidor con su nombre, dentro de la cual hay 4 ficheros PHP similares en cada caso.



El servidor dispone de un fichero .htaccess, donde he declarado una serie de directivas que permiten que, a la hora de llamar a cualquier parte del api, se pueda obviar la extensión del fichero, por ejemplo, se puede llamar a /api/notes/delete, en vez de llamar a /api/notes/delete.php, siendo más cómodo realizar peticiones a la API.

/public_html/.htaccess

```
1
2 # HTID:21560761: DO NOT REMOVE OR MODIFY THIS LINE AND THE LINES BELOW
3 php_value display_errors 1
4 # DO NOT REMOVE OR MODIFY THIS LINE AND THE LINES ABOVE HTID:21560761:
5 Header add Access-Control-Allow-Origin "*"
6 Header add Access-Control-Allow-Methods: "GET,POST,OPTIONS,DELETE,PUT"
7
8 RewriteEngine On
9 RewriteCond %{REQUEST_FILENAME} !-d
10 RewriteCond %{REQUEST_FILENAME} \.php -f
11 RewriteRule ^(.*)$ $1.php [NC,L]
12 #RewriteRule ^api/([a-zA-Z0-9]+)/([0-9]+)/?$ api/$1.php?id=$2 [NC,L]
```

Tomando como referencia que estamos dentro de la carpeta “notes”, donde se manejaría la lógica de datos para que el usuario tenga un tablero con notas, la estructura y funcionalidad de cada fichero sería la siguiente:

Get.php

En este fichero se encuentra la lógica para buscar y traer todas las notas almacenadas de un usuario, en este caso. Se conecta con la base de datos y realiza una sentencia SELECT, donde obtiene de la tabla “notas” todas aquellas notas que pertenezcan al usuario X, utilizando el campo “owner” para realizar el filtro, el cual equivale al ID del usuario que hace la petición.

Cada vez que una funcionalidad carga, esta es la primera sentencia que se ejecuta, puesto que, si el usuario accede a su panel de notas, quiere ver todas las notas que tiene, por ello nada más se carga el componente de Vue referente con el tablero de notas, se hace una petición HTTP/GET a este “endpoint” de la API para obtener los datos necesarios.

1. Se conecta con la base de datos.
2. Si el usuario le ha pasado un ID de nota, realiza una sentencia SELECT filtrando por ID, por lo que solo devuelve una nota. En cambio, si no recibe un parámetro ID, devolverá todas las notas de ese usuario (este suele ser el caso más común).

```
if (isset($_GET['user_id'])){
    $result = $conn->query("SELECT * FROM notes WHERE owner = '".$_GET['user_id']."'");
}else{
    $result = $conn->query("SELECT * FROM notes");
}
```

3. Si encuentra notas para ese usuario, devolverá todas las notas en un solo objeto en formato JSON, de lo contrario devolverá “false” y Vue se encargará de reaccionar e indicar que ese usuario no tiene notas.

```
if ($result->num_rows > 0) {
    $data = array();
    while ($row = $result->fetch_assoc()) {
        $data[] = $row;
    }
    $json = json_encode($data);

    header("Content-Type: application/json");
    echo $json;
} else {
    header("Content-Type: application/json");
    echo json_encode(false);
}
```

Add.php

En este fichero se encuentra la lógica para añadir una nueva nota en este caso. Se conecta con la base de datos y realiza una sentencia INSERT, donde inserta a la tabla “notas” un nuevo registro que todos los datos que ha recibido del usuario, y lo guarda usando el campo “owner”, donde se indica el ID del usuario al que le pertenece esta nueva nota.

1. Se conecta con la base de datos.
2. Recoge el objeto que le ha pasado el usuario a través de la petición HTTP/POST:

```
header("Content-Type: application/json; charset=UTF-8");
$json = file_get_contents('php://input');
$data = json_decode($json);

$title = $data->title;
$content = $data->content;
$date = $data->date;
$color = $data->color;
$owner = $data->owner;
```

3. Prepara la sentencia INSERT de SQL y la ejecuta con los datos correspondientes, teniendo en cuenta de crear la nota para el usuario que lo ha pedido:

```
// prepare the SQL statement
$stmt = $conn->prepare("INSERT INTO `notes`(`title`, `content`, `date`, `color`, `owner`) VALUES (?, ?, ?, ?, ?)");
$stmt->bind_param("sssss", $title, $content, $date, $color, $owner);
```

4. Si todo ha ido correctamente, le devuelve la nueva nota creada de la BD como objeto JSON a Vue, para que este lo muestre en la interfaz de usuario y el usuario reciba el feedback de que la nota ha sido creada. En caso de fallar, devolvería un mensaje de error.

```
if ($stmt->execute()) {
    $noteId = $stmt->insert_id;
    $note = array(
        "note_id" => $stmt->insert_id,
        "title" => $title,
        "content" => $content,
        "date" => $date,
        "color" => $color,
        "owner" => $owner
    );
    http_response_code(200);
    echo json_encode($note);
} else {
    http_response_code(500);
    echo json_encode(array("message" => "Error creating note: " . $stmt->error));
}
```

Edit.php

En este fichero se encuentra la lógica para editar una nueva nota, en este caso ya existente. Es muy parecido a añadir, pero modificando la “Query” o sentencia a ejecutar. Se conecta con la base de datos y realiza una sentencia UPDATE, donde actualiza en la tabla “notas” un registro ya existente del usuario, y lo modifica usando el campo “owner”, donde se indica el ID del usuario al que le pertenece esta nota.

1. Se conecta con la base de datos.
2. Recoge el objeto que le ha pasado el usuario a través de la petición HTTP/PUT:

```
header("Content-Type: application/json; charset=UTF-8");
$json = file_get_contents('php://input');
$data = json_decode($json);

$noteId = $data->note_id;
$title = $data->title;
$content = $data->content;
$date = $data->date;
$color = $data->color;
$owner = $data->owner;
```

3. Prepara la sentencia UPDATE de SQL y la ejecuta con los datos correspondientes, teniendo en cuenta de actualizar la nota con el ID recibido y para el usuario que lo ha pedido:

```
$stmt = $conn->prepare("UPDATE `notes` SET `title`=?, `content`=?, `date`=?, `color`=?, `owner`=? WHERE `note_id`=?");
$stmt->bind_param("sssssi", $title, $content, $date, $color, $owner, $noteId);
```

4. Si todo ha ido correctamente, le devuelve la nueva nota con los datos actualizados como objeto JSON a Vue, para que este lo muestre en la interfaz de usuario y el usuario reciba el feedback de que la nota ha sido actualizada. En caso de fallar, devolvería un mensaje de error.

```
if ($stmt->execute()) {
    $note = array(
        "id" => $noteId,
        "title" => $title,
        "content" => $content,
        "date" => $date,
        "color" => $color,
        "owner" => $owner
    );
    http_response_code(200);
    echo json_encode($note);
} else {
    http_response_code(500);
    echo json_encode(array("message" => "Error updating note: " . $stmt->error));
}
```

Delete.php

En este fichero se encuentra la lógica para borrar una nueva nota, en este caso. Se conecta con la base de datos y realiza una sentencia DELETE, donde elimina el registro de la tabla “notas” usando el campo “owner”, donde se indica el ID del usuario al que le pertenece esta nota y el campo ID que recibe en la petición, para eliminar solo la nota que tiene ese ID específico.

1. Se conecta con la base de datos.
2. Recibe un ID como parámetro en la petición HTTP/DELETE. Si el parámetro viene vacío, saltará un error indicando que no puede ser vacío, en caso de que el ID llegue correctamente, se ejecutará una sentencia DELETE donde se filtrará por el ID y se eliminará de la base de datos.

```
if (isset($_REQUEST['id'])) {  
    $stmt = $conn->prepare("DELETE FROM `notes` WHERE `note_id`=?");  
    $stmt->bind_param("i", $_REQUEST['id']);  
  
    if ($stmt->execute()) {  
        http_response_code(200);  
        echo json_encode(array("message" => "Note deleted successfully"));  
    } else {  
        http_response_code(500);  
        echo json_encode(array("message" => "Error deleting note: " . $stmt->error));  
    }  
} else {  
    http_response_code(500);  
    echo json_encode(array("message" => "No ID was provided"));  
}
```

Documentación online de Mee

La documentación de la API de Mee se encuentra alojada en la página de GitBooks. En dicha documentación, se pueden encontrar todos los “**endpoints**” que tiene disponibles la API.

La documentación se puede encontrar en el siguiente enlace:

<https://mee-3.gitbook.io/mee-api-documentation/reference/api-reference>

7. Conclusión

Una vez finalizado el desarrollo de la aplicación web “Mee”, es conveniente mirar hacia atrás y valorar los resultados que se han obtenido, mencionar aquellos puntos o cosas que por ciertos motivos han quedado pendientes y realizar una autoevaluación personal del trabajo realizado, así como calcular el tiempo y esfuerzo dedicado al desarrollo e investigación.

7.1 Resultados obtenidos

Después de haber terminado de desarrollar la aplicación web Mee, creo que el resultado es el que esperaba, pues la idea de la aplicación ya estaba planteada, incluso la interfaz visualmente ya quedó plasmada en la aplicación móvil del año pasado.

El objetivo de este año era rehacer y adaptar esa interfaz a pantallas más grandes como tabletas y ordenadores, pero manteniendo la interfaz de móvil simple y amigable, de forma que Mee estuviera disponible tanto en teléfonos móviles en formato app, como en ordenadores y cualquier otro dispositivo más grande en formato web.

Aunque el objetivo era conectar la aplicación web con la aplicación móvil, no se ha podido completar al 100%, puesto que el desarrollo del año pasado fue muy interno y dependiente, sin base de datos externa. Sin embargo, este año he aprendido de los errores y he creado una aplicación web escalable, que en cualquier momento podría adaptarse a cambios o nuevas funciones, e incluso para conectarse con una versión de móvil haciendo uso de la base de datos compartida.

Creo que el resultado es muy bueno, y aunque no he cumplido con todo lo que me gustaría, ha quedado muy bien, he conseguido que sea muy parecida a la app móvil, y sobre todo he mejorado cosas que el año pasado no fui capaz de ver, por lo que estoy contento de que el robot Mee pueda ayudar a más y más gente, sin importar el dispositivo que esté usando.

7.2 Puntos pendientes

Durante el proceso de planteamiento del proyecto, tenía claro que quería expandir el legado del robot Mee más allá de los dispositivos Android. Durante mucho tiempo, pensé en adaptar la aplicación móvil que desarrollé el año pasado en una mucho más universal, que se pudiera ejecutar desde cualquier dispositivo con conexión a Internet.

Entonces surgió la idea de adaptar “Mee” a una aplicación web, conservando la mayoría de las funcionalidades posibles.

La idea era hacer las mismas funcionalidades, pero algunas no podrían ser desarrolladas de la misma manera, pues por ejemplo las alarmas de tu dispositivo móvil no se pueden establecer de igual forma en un ordenador, o los calendarios son distintos, o incluso los accesos directos que tiene la aplicación de móvil no tienen sentido en el ordenador, pues no existen esas aplicaciones en el ordenador de la misma forma.

Es por ello por lo que he optado por mantener las máximas funcionalidades posibles, descartando aquellas que son más difíciles y algunas que por falta de tiempo no he podido rehacer de la misma forma.

Compatibilidad web-móvil

Uno de los puntos pendientes más importantes ha sido la compatibilidad directa entre aplicación móvil y aplicación web, pues en un principio quería que todas las acciones que hiciera el usuario en la aplicación móvil también se pudieran consultar desde la web y viceversa, haciendo que la misma información sea accesible y modificable desde ambos lugares.

Sabía que sería complicado, pero realmente cuando empecé a desarrollar la aplicación web me di cuenta de que no iba a ser posible, pues el desarrollo de la aplicación de Android fue muy concreto y limitado, y la información que creaba el usuario (notas, listas de la compra, etc.) se guardaban en la memoria del teléfono, y no en una base de datos externa, de forma que era imposible conectar ambas aplicaciones en un único ecosistema de datos.

De igual forma, al ser un desarrollo nativo de Android, aquellos usuarios que hicieran uso de otros sistemas operativos como iOS no tendrían acceso a la versión móvil a menos que se desarrollara una versión específica para estos dispositivos.

Los usuarios también fueron un problema, pues en la app móvil del año pasado no era necesario guardar información del usuario ni tener un sistema de login y registro, pues la información del usuario se guardaba en su teléfono y no había varias personas compartiendo datos. En el caso de la web esto era un problema, pues era necesario tener una tabla de usuarios con su correo y contraseña para que cada usuario pudiera solo consultar sus datos, y no los de nadie más.

El propósito de este proyecto no era conectar las aplicaciones, sino crear la aplicación web de forma independiente, y si hubiera sido posible, entonces conectarlas. Es por ello por lo que a pesar de no haber sido posible conectarlas, mantuve la idea y continué con el desarrollo de la versión web, esta vez teniendo en cuenta posibles integraciones en el futuro.

A pesar de que se ha quedado como punto pendiente la interconexión de ambas aplicaciones, esta vez he tenido en cuenta futuras posibilidades y he adaptado la aplicación web para que guarde los datos en una base de datos externa, manteniendo la mayoría de campos que guardaba la aplicación móvil, y si en el futuro se desarrolla una nueva versión para la app móvil, la misma base de datos podría ser usada tanto para la web como para el móvil, de forma que sería mucho más sencillo crear ese ecosistema de datos.

Algunas funciones nuevas y otras actualizaciones por desgracia han quedado en el tintero, pues lamentablemente el desarrollo de una aplicación web es costoso y requiere mucho tiempo y pruebas de error, por lo que, con el tiempo limitado y extensión de todo el contenido relacionado con el proyecto, todas estas funciones tan interesantes no han podido ser implementadas, y ciertamente Mee sigue un poco triste por no poder ayudar a todas estas personas...

Sin embargo, es importante reconocer que, aunque algunas funciones no han podido ser implementadas, el trabajo realizado ha sido importante y reconocible, por lo que en un futuro siempre se puede actualizar y añadir nuevas funciones que no se hayan podido implementar ahora.

Mee no descansará hasta que consiga cumplir sus objetivos y pueda ayudar a que todo el mundo pueda disfrutar de las funciones de organización personal que ofrece el simpático robot Mee.

7.3. Tiempo dedicado y dificultad

El proceso de desarrollo de Mee Web ha sido, en muchos momentos, divertido y educativo, pues me lo he pasado muy bien y he aprendido muchas cosas nuevas durante el desarrollo, pero por supuesto también ha habido muchos momentos difíciles y de frustración, pero he sabido superar todos los errores que han ocurrido y por supuesto he investigado cómo solucionarlos y he aprendido mucho de los errores.

En general, el tiempo de desarrollo total habrá sido de unas 80-100 horas aproximadamente, puesto que es una web con muchas funcionalidades, cada una de ellas totalmente independiente del resto y con su lógica y programación concreta. Además, gran parte de ese tiempo se ha dedicado a maquetación y diseño HTML y CSS, ya que el objetivo era que la interfaz fuera casi exacta a la aplicación móvil, y el diseño es 100% único y empezado desde cero, sin usar plantillas.

Algunas funcionalidades de la aplicación han resultado sencillas de desarrollar, pues su complejidad no era demasiado alta, véase la función de notas, por ejemplo, pues era un concepto sencillo, tanto lógicamente como funcionalmente.

En cambio, otras funciones, a pesar de ser sencillas, han resultado ser más complicadas debido a recursos externos o a la necesidad de adquirir ciertos conocimientos e investigación al respecto. Véase por ejemplo la función de noticias era teórica y visualmente sencilla, pero la parte de desarrollo interno, de cómo se buscan y obtienen las noticias y cómo funcionan las diferentes acciones para cambiar entre categorías han sido un reto mucho mayor, pues no solo era la lógica sino también encontrar una API que fuera gratuita y que proporcionase la información necesaria para las necesidades de la funcionalidad. Además, la API de noticias que usé para la aplicación móvil ha cambiado sus políticas durante el desarrollo y he tenido que buscar una parecida y volver a adaptar el código.

Otras funciones han pasado por varias fases o estados, pues el resultado final no siempre es igual que el resultado inicial, y por ciertas circunstancias, algunas funcionalidades se modifican durante el desarrollo, algunas para agregar cosas nuevas y otras eliminando alguna funcionalidad que no encaje o no sea relevante.

Por ejemplo, la adaptación de la pantalla principal o pantalla Home ha pasado por varias fases distintas durante el desarrollo. En la aplicación de Mee original, la pantalla de Home era una lista de accesos directos a aplicaciones instaladas en el teléfono móvil, pero en el entorno web, esa misma funcionalidad no es posible de la misma forma, pues todas las aplicaciones instaladas en el móvil son de móvil, por lo que no tiene sentido mantener esa funcionalidad si no es posible abrir esos accesos directos a las aplicaciones desde el navegador web.

En general, algunas funcionalidades han sido más fáciles, otras más difíciles, pero en general la dificultad real del proyecto ha sido realizar cada una de las funciones por separado, cada una con su propio planteamiento y desarrollo, y luego juntar todas las funciones para que funcionen como una sola aplicación web.

Además, hay desarrollos que no se ven a simple vista por el usuario, pero que realmente son indispensables en la aplicación, como por ejemplo la base de datos donde se almacenan todos los datos de los usuarios, y la API que utiliza la base de datos para extraer los datos del usuario en cada momento que sea necesario. Estos desarrollos también tienen su propio desarrollo y dificultad, que se añaden al total.

7.4. Valoración personal y autoevaluación

Considero que realizar un proyecto de esta magnitud es realmente un reto único para el alumno, pues de normal realizamos proyectos pequeños e independientes, pero un proyecto como este requiere mucho trabajo, tanto de desarrollo como de planteamiento y organización. Esto permite al alumno utilizar todos los recursos que ha adquirido durante el año actual y años anteriores e incluso permite una evolución de la forma de trabajo, pues debe adaptarse a un periodo de tiempo limitado y una serie de pautas a seguir.

Creo que mi trabajo ha sido muy divertido de realizar, pues es una aplicación web que yo mismo he planteado, y que llevaba cierto tiempo queriendo desarrollar, por lo que poder empezar el desarrollo y aún más importante terminarlo, es una experiencia gratificante y he aprendido mucho durante el desarrollo, tanto conceptos nuevos como actualizar antiguos conceptos y aprender de los errores.

También me entristece no haber podido realizar todas las cosas que me hubiera gustado hacer, pues inicialmente tenía un par de funcionalidades pensadas adicionales, no demasiado importantes, pero que hubieran aportado cierto valor al proyecto.

Sin embargo, estoy muy contento con el trabajo realizado, y lo más importante a destacar creo que es el hecho de haber sido capaz de completar todos los objetivos que tenía planeados en el periodo de tiempo establecido.

8. Bibliografía y Webgrafía

- Stack Exchange. Inc (2023). Stack Overflow. Recuperado de <https://stackoverflow.co>
- Weather API (2023). Weather API developers. Recuperado de <https://www.weatherapi.com>
- News API (2023). NewsAPI developers. Recuperado de <https://newsapi.org>
- GNews API (2023). GNewsAPI developers. Recuperado de <https://gnews.io/>
- Free Dictionary API (2023). Dictionaryapi. Recuperado de <https://dictionaryapi.dev/>
- Rapid API (2023). Microsoft Azure. Recuperado de <https://rapidapi.com/microsoft-azure-org-microsoft-cognitive-services/api/microsoft-translator-text/>
- Google Inc. (2023). Google Maps. Recuperado de <https://maps.google.com/maps>
- Nominatim StreetMaps (2023). OpenStreetMaps. Recuperado de <https://nominatim.openstreetmap.org>
- OpenAI Inc. (2023). ChatGPT by OpenAI. Recuperado de <https://chat.openai.com/>
- Henry Boisdequin (2020). React vs Vue vs Angular vs Svelte . Recuperado de <https://dev.to/hb/react-vs-vue-vs-angular-vs-svelte-1fdm>

9. Recursos utilizados

Para el desarrollo de la aplicación de Mee se han necesitado distintas herramientas, la gran mayoría de ellas programas o “software” relacionadas con el desarrollo. También he usado herramientas de edición de imagen y herramientas de inteligencia artificial para código e imágenes. A continuación, se listan las herramientas usadas más importantes para el desarrollo:

- **Visual Studio Code (VSCode):** Herramienta fundamental y la más usada en todo el proyecto, pues es el IDE (entorno de desarrollo) utilizado para crear la web al completo, tanto interfaz como la funcionalidad y complementos de esta.
- **Photoshop y variantes:** Utilizado para el tratado y confección de imágenes personalizadas para darle vida al robot virtual protagonista, conocido como Mee.
- **Microsoft Word y Microsoft Power Point:** Utilizado para la confección de la memoria de la presentación del proyecto.
- **Weather API:** Utilizado para la obtención de los datos meteorológicos.
- **News API:** Utilizado para obtener la información y las noticias del apartado de noticias *(Tuve que cambiarla en mitad de desarrollo por cambio de políticas en su plan gratuito)*.
- **GNews API:** Utilizado para obtener la información y las noticias del apartado de noticias.
- **Github / Github Pages:** Utilizado como control de versiones de todo el código del proyecto. Además, la web se ha subido a Internet haciendo uso de una de sus aplicaciones llamada GitHub Pages.
- **000webhost by Hostinger:** Utilizado para “hostear” la API en PHP y Base de Datos donde se almacenan y extraen todos los datos de la aplicación web.

10. Glosario

10.1. Siglas

- **IA:** La inteligencia artificial es la unión de diferentes algoritmos de programación cuyo objetivo es crear máquinas o robots que imiten y ejecuten las tareas y capacidades de un ser humano, con el propósito de ser utilizadas para tareas complejas o repetitivas.
- **IDE:** El entorno de desarrollo integrado (Integrated Development Enviroment) son programas o “software” utilizados por programadores y desarrolladores para facilitar la creación y desarrollo de aplicaciones y otro tipo de software. Estos IDE proporcionan al desarrollador las herramientas básicas y avanzadas para facilitar ciertas tareas, así como ciertas ayudas (autocorrección de palabra, sugerencias, indicador de errores, etc.)
- **API REST:** Una Interfaz del Programa de Aplicación (Aplication Programming Interface) es un conjunto de reglas que permite que varios programas se comuniquen entre sí y se transfieran datos de uno al otro. Para ello los datos se almacenan en un servidor, el cual permite a varios clientes extraer esos datos a través de llamadas HTTP.
- **AJAX:** Asynchronous JavaScript and XML es una técnica de desarrollo web que permite la comunicación asíncrona entre un servidor y un cliente, sin necesidad de recargar completamente la página. Se basa en el uso conjunto de tecnologías como JavaScript y XML.
- **SPA:** "Single Page Application" (Aplicación de una sola página). Es un tipo de aplicación web que carga una única página inicial y luego actualiza el contenido dinámicamente sin recargar la página completa.

10.2. Terminología

- **Framework:** Proporciona estructura y herramientas para el desarrollo de aplicaciones, simplificando tareas comunes y fomentando la reutilización de código.
- **Streaming:** El proceso de reproducción continua de contenido multimedia o datos en tiempo real sin necesidad de descargarlo por completo antes de reproducirlo.
- **TypeScript:** Un lenguaje de programación de código abierto que amplía la sintaxis de JavaScript al agregarle tipos estáticos opcionales, lo que permite detectar y prevenir errores durante la etapa de desarrollo.
- **(API) endpoint:** Un punto final (URL) expuesto por una API (Interfaz de Programación de Aplicaciones) que permite a los clientes enviar solicitudes y recibir respuestas para acceder recursos específicos de la API.
- **Query (Sentencia):** Una declaración o instrucción escrita en el lenguaje SQL (Structured Query Language) que se utiliza para recuperar, manipular o modificar datos en una base de datos.
- **Aplicación Web:** Una aplicación accesible a través de un navegador web, que no requiere instalación y puede ser utilizada en diferentes dispositivos

- **Aplicación nativa:** Una aplicación diseñada específicamente para un sistema operativo o dispositivo en particular, aprovechando al máximo las capacidades y funcionalidades de este.
- **Escalabilidad:** La capacidad de un sistema o aplicación para adaptarse y manejar eficientemente un aumento en la carga de trabajo o en el número de usuarios. Se divide en horizontal y vertical.
- **Lenguaje de scripting:** Un lenguaje de programación ligero y de alto nivel utilizado para escribir scripts, generalmente para la automatización de tareas básicas o repetitivas.

11. Resumen del contenido

La organización es una de las cosas más importantes en de día a día de una persona, pues una buena organización de tareas y tiempo permite a la persona aprovechar mejor todo el tiempo del que dispone y así ser más productivo. Es por ello por lo que tener algo a alguien que nos ayude con la organización puede ser beneficioso.

Mee es un pequeño robot que ayudará a cualquier persona que utilice tanto su aplicación web como móvil, a gestionar su tiempo y sus tareas diarias, así como ofrecerle diferentes métodos de ocio y obtención de información.

Para ello, Mee dispone de varias herramientas con las que ayudar y se encarga de centralizar todas las cosas que una persona utiliza en cualquier dispositivo con acceso a navegador e internet en su día a día en una sola aplicación, para mejorar el proceso de organización del usuario.

El objetivo es minimizar el tiempo y esfuerzo de realizar tareas como consultar noticias, el tiempo, diccionarios... y organizar las tareas, recordatorios y notas del usuario.

En conclusión, cualquier persona que necesita cierta ayuda para poder organizarse mejor, o bien le interese la idea de tener todas las funciones en un mismo lugar, ¡Mee le ayudará en cualquier cosa como forma de agradecimiento por confiar en él!